ISSN 0265-2919
80p
21

THE HOME COMPUTER ADVANCED COURSE

MAKING THE MOST OF YOUR MICRO

An ORBIS Publication
IR £1 Aus $1.95 NZ $2.25 SA R1.95 Sing $4.50 USA & Can $1.95

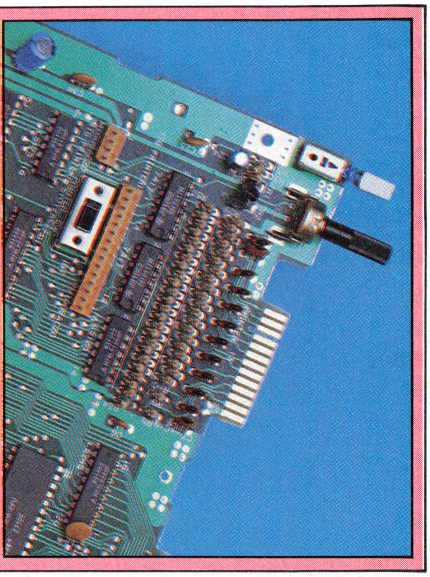
CONTENTS

Next Week

We discover how computers are designed and assembled. Often a single computer will be put together in several different countries: from design in America, to chip manufacture in Singapore and assembly in Taiwan.

We put the Amstrad computer through its paces. This interesting home computer comes complete with its own monitor and has a larger memory and a lower price tag than the BBC Micro.

The VTX5000 is a custom-designed modem for the Spectrum that can link you to Prestel and Micronet information services.



QUIZ

1) The EP-44 uses a thermal printer, therefore special thermal paper has to be used with the machine. True or False?

2) How is the strength of a shot determined by using the keyboard in Snooker?

3) The X co-ordinate on the BBC Micro can reach a maximum of 1,280. However, this cannot be used by the programmer for plotting. Explain why.

4) What is the difference between a decision tree and a decision table?

Answers To Last Week's Quiz

A1) Source code is written by the programmer, object code is the machine language used by the computer.

A2) As mode 5 has only four colours, each requiring two bits, four pixels can be represented.

A3) The user will have to load the cassette-based SmartBASIC.

A4) The Acme Interstellar Transport Company.

QUIZ

COVER PHOTOGRAPHY BY MARCUS WILSON-SMITH

Editor Jim Lennox; **Art Director** David Whelan; **Technical Editor** Brian Morris; **Production Editor** Catherine Cardwell; **Art Editor** Claudia Zeff; **Chief Sub Editor** Robert Pickering; **Designer** Julian Dorr; **Art Assistants** Liz Dixon, Gary Capps-Jenner; **Editorial Assistant** Stephen Malone; **Sub Editor** Steve Mann; **Researchers** Melanie Davis, Martha Ellen Zenfell; **Contributors** Steve Colwill, Steve Malone, Geoff Nairn, Geoff Bains, Matt Nicolson, Richard Pawson; **Group Art Director** Perry Neville; **Managing Director** Stephen England; **Published by Orbis Publishing Ltd: Editorial Director** Brian Innes; **Project Development** Peter Brookesmith; **Executive Editor** Chris Cooper; **Production Controller** Peter Taylor-Medhurst; **Circulation Director** David Breed; **Marketing Director** Michael Joyce; **Designed and produced by Bunch Partworks Ltd; Editorial Office** 85 Charlotte Street, London W1P 1LB; **© APSIF Copenhagen 1984; © Orbis Publishing Ltd 1984: Typeset by Universe; Reproduction by Mullis Morgan Ltd; Printed in Great Britain by Artisan Press Ltd, Leicester**

HOME COMPUTER ADVANCED COURSE – Price UK 80p IR £1.00 AUS $1.95 NZ $2.25 SA R1.95 SINGAPORE $4.50 USA and CANADA $1.95

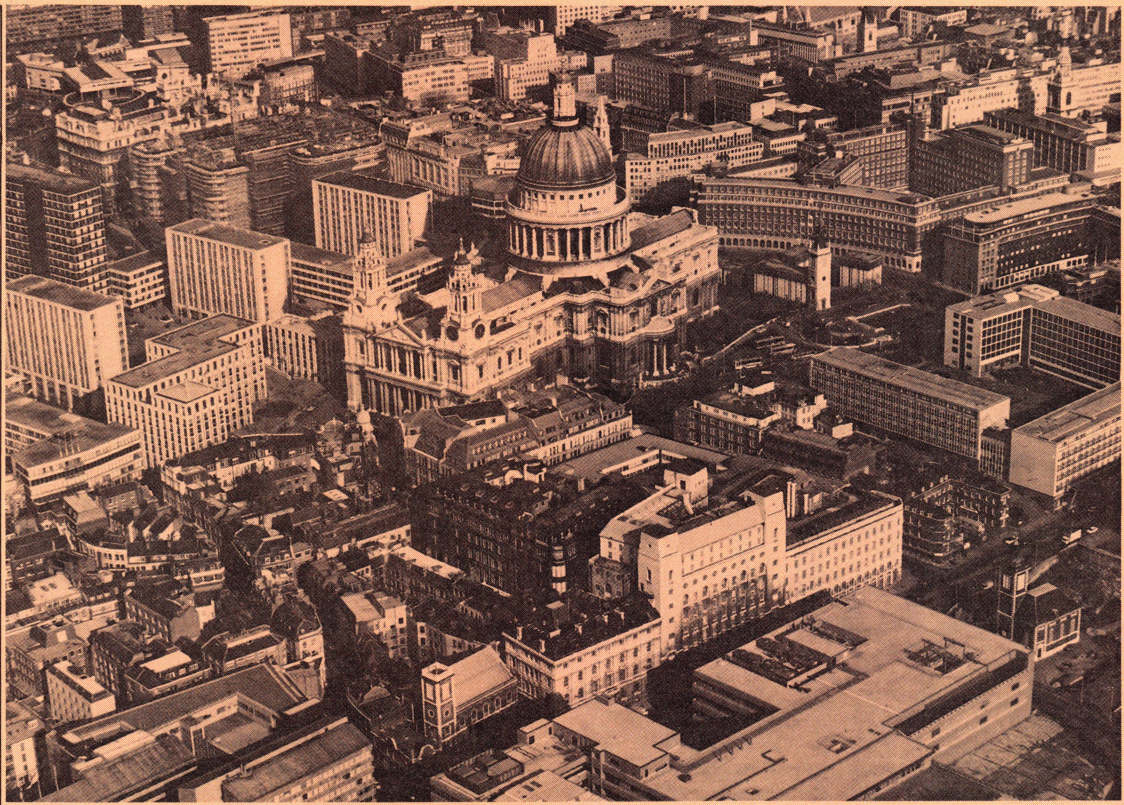
How to obtain your copies of HOME COMPUTER ADVANCED COURSE – Copies are obtainable by placing a regular order at your newsagent, or by taking out a subscription. Subscription rates: for six months (26 issues) £23.80; for one year (52 issues) £47.60. Send your order and remittance to Punch Subscription Services, Watling Street, Bletchley, Milton Keynes, Bucks MK2 2BW, being sure to state the number of the first issue required.

Back Numbers UK and Eire – Back numbers are obtainable from your newsagent or from HOME COMPUTER ADVANCED COURSE. Back numbers, Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT at cover price. AUSTRALIA: Back numbers are obtainable from HOME COMPUTER ADVANCED COURSE. Back numbers, Gordon & Gotch (Aus) Ltd, 114 William Street, PO Box 767G Melbourne, Vic 3001. SOUTH AFRICA, NEW ZEALAND, EUROPE & MALTA: Back numbers are available at cover price from your newsagent. In case of difficulty write to the address in your country given for binders. South African readers should add sales tax.

How to obtain binders for HOME COMPUTER ADVANCED COURSE – UK and Eire: Please send £3.95 per binder if you do not wish to take advantage of our special offer detailed in Issues 5, 6 and 7. EUROPE: Write with remittance of £5.00 per binder (incl. p&p) payable to Orbis Publishing Limited, 20/22 Bedfordbury, LONDON WC2N 4BT. MALTA: Binders are obtainable through your local newsagent price £3.95. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Miller (Malta) Ltd, M.A. Vassalli Street, Valletta, Malta. AUSTRALIA: For details of how to obtain your binders see inserts in early issues or write to HOME COMPUTER ADVANCED COURSE BINDERS, First Post Pty Ltd, 23 Chandos Street, St. Leonards, NSW 2065. The binders supplied are those illustrated in the magazine. NEW ZEALAND: Binders are available through your local newsagent or from HOME COMPUTER ADVANCED COURSE BINDERS, Gordon & Gotch (NZ) Ltd, PO Box 1595, Wellington. SOUTH AFRICA: Binders are available through any branch of Central Newsagency. In case of difficulty write to HOME COMPUTER ADVANCED COURSE BINDERS, Intermag, PO Box 57394, Springfield 2137.

Note – Binders and back numbers are obtainable subject to availability of stocks. Whilst every attempt is made to keep the price of the issues and binders constant, the publishers reserve the right to increase the stated prices at any time when circumstances dictate. Binders depicted in this publication are those produced for the UK market only and may not necessarily be identical to binders produced for sale outside the UK. Binders and issues may be subject to import duty and/or local taxes, which are not included in the above prices unless stated.

FINANCIAL MICRO TIMES



Power to the People

Many people relish the idea of running their own business. But even if it's only a small shop, the problems associated with such a venture can be formidable. And nowhere is this more so than when dealing with your company's finances. However, this article is not about practical business software, but is dedicated to a selection of games software that simulates some of the problems and challenges of big business. The best-known of such games is Monopoly, which is adapted from the board game of property wheeling and dealing. Business games are ideally suited to computers, since there are no tokens or paper money to lose and the computer handles all the calculations.

The range of companies that you can run is quite diverse: breweries, airlines, car firms and farms are just a few examples. Each type of concern obviously has its own problems. The price of oil is very important to an airline, and there are problems common to all, such as inflation or high loan rates.

No matter what type of company you choose, business simulation programs work in much the same way. You are invariably manager of a particular company and you have an initial amount of cash at the beginning of the game. In the Corn Cropper program, in which you are running a farm, this is just £50,000, but if you are in charge of an oil company, as in the Dallas game, you need $100 million just to get off the ground.

You then invest some of this capital in what are called 'fixed assets' — aeroplanes, car factories, land, etc. Using these, you can then start producing your goods or services — flights, cars, wheat — and earn some income. Obviously, it's not quite as simple as that and your success or failure as a company depends on many other factors.

The degree of realism varies from program to program. Some, like Dallas, are obviously intended mainly for amusement. Others can be very realistic: the Corn Cropper program has data on sunshine and rainfall, ideal conditions for growing wheat, salaries of hired labourers and running costs for tractors — just a few of the factors that modern farmers must consider.

There will inevitably be some people who are not satisfied with merely running a company — not even a $100 million oil corporation. For these people, the obvious answer is to try running a country! 1984 is a game that simulates the economy of Great Britain, with you as the Prime Minister.

To simulate the national economy, economists use a 'model' — not a physical model, but rather a complex diagram showing how all the parts of the economy interact with each other. The model in the 1984 program is very simple, containing just five sections — the Government, the banks, the population, industry and the rest of the world. Even so, there are at least 30 inter-connections in this model — government grants to industry, income tax paid by the population to the Government, and so on.

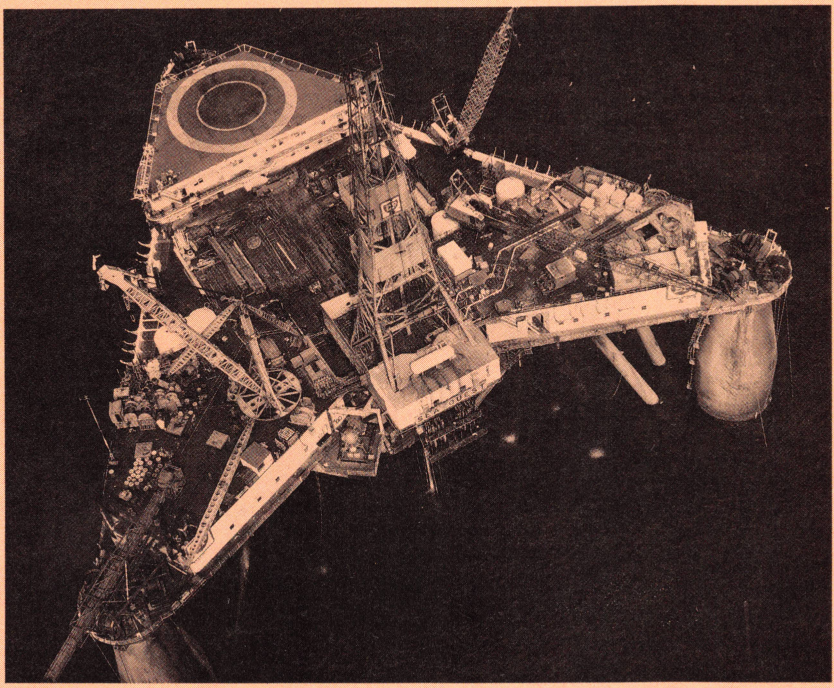
The Treasury also has an economic model, which the British Government uses to make various predictions about the national economy. As can be imagined, this model is a little more complicated. The simulation program, which contains the model, is called AMODEL and runs on a Sperry 1100 mainframe computer. To make its predictions it uses 1,100 different economic variables — inflation rate, unemployment, etc., collected over the past 10 years. This data alone occupies about 250 Kbytes of memory, but when the program is running at full capacity nearly eight Mbytes are used.

Typically the model can be used to predict unemployment figures, the level of imports and exports, the exchange rate for the dollar and many other economic indicators. To give you an idea of how complex the model is, the task of forecasting next year's inflation rate would take two or three minutes of actual processor time. Not long you might think, until you bear in mind that the processor handles about 10 million machine code instructions in just one second. Printing out the forecast takes another 15 or 20 minutes.

Computer simulation games offer you the chance to run your own business — or even to run a whole nation's economy — without the risk of losing your life savings if the company goes bankrupt. Unfortunately you don't enjoy the profits either!

FINANCIAL MICRO TIMES

STRIKING OIL



ASSOCIATED PRESS

Dallas is a business game that puts you at the head of your own oil company. Starting out with $100 million in cash, you have to try and accumulate $200 million in order to take over your biggest rival — Euing Associates. (The mis-spelling of the name from the popular television series is intentional.)

On the screen is a map of Texas, and as the game progresses you have the chance to gain 'concessions' in each of the squares. To win the concession you have to put in a successful bid, but once you have made one you can then drill a test well to search for oil. If oil is discovered in your field, you then set up a drilling rig and a production facility, and finally a pipeline.

If you overcommit yourself — by buying up too many concessions before you have any oil revenue coming in, for example — then you might have to go to the bank for a loan, and then you start paying interest. If the loan exceeds $20 million, then you risk being taken over by Euing Associates.

This program is available from Cases Computer Simulations for a range of home micros, including the BBC Micro, Spectrum, Oric and Electron.

HIGH FINANCE

In this game you are chairman of L-AIR, a private airline with capital of £3 million at the beginning of the game. Within seven years, you must increase your assets to £30 million, at which point you can take over British Airways and win the game.

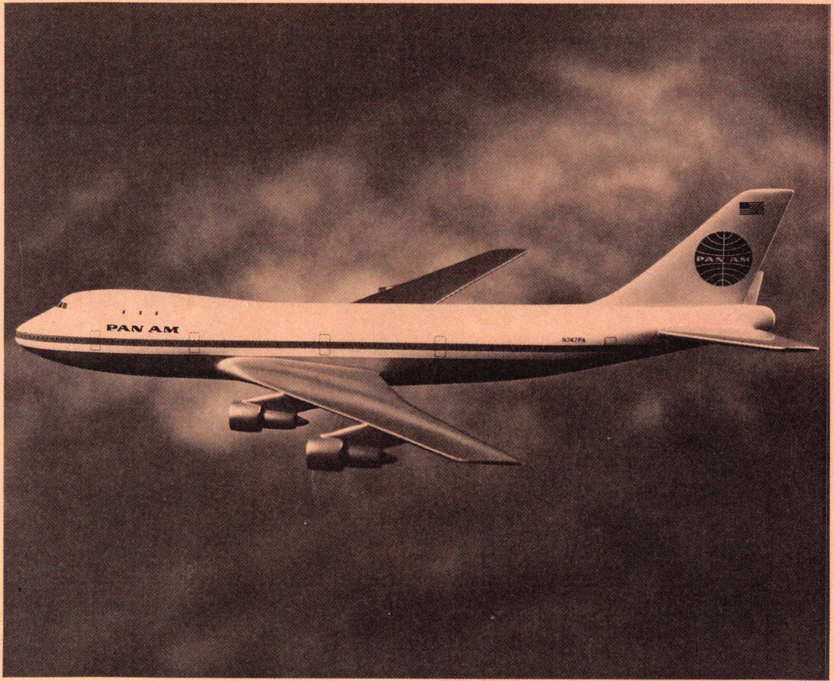
At the beginning of each financial year you need to decide how many aircraft to operate, using forecasts that predict the number of passengers. Initially, you do not have enough money to buy an aeroplane — they cost £10 million each — so you have to rent them. In more profitable years you will need to decide whether it is better to hire or buy aircraft, and this depends on both charter rates and loan interest rates.

You also have to decide on the level of manning and maintenance for your aircraft. If it's too low then you may not have enough staff or planes and some flights might have to be cancelled. If it is too high, you will waste money.

Occasionally a 'ticker tape' crosses the screen containing various telex messages. For example, OPEC might have increased the price of oil — and therefore aviation fuel — or you might have been granted a licence for a new air route.

At the end of the year a balance sheet gives L-AIR's overall performance and a chairman's statement on how successful the company has been. If you lose £10 million in the first year, then the company is liquidated.

Airline is available for the Spectrum, Oric, Electron and BBC Micro from Cases Computer Simulations.



ASSOCIATED PRESS

Prime Opportunity

ASSOCIATED PRESS

In this game you are the Prime Minister of Great Britain in the year 1984. Your aim is to stay in office for as long as possible, and your popularity is determined by your success in balancing the country's books.

At the beginning of your term you have to make various financial decisions. To help you there are eight economic indicators, including the rate of inflation and unemployment, etc. At the beginning of each year a graph shows how any one indicator has changed during your years in office.

Your first decision is to determine the minimum lending rate, which sets the interest rate for the year. You then have to negotiate the pay increases for the Civil Service, the Public Sector and the Private Sector. If you give too extravagant a pay settlement you will be asked to resign!

After the wage settlements, you must decide on the levels of spending by the various Government departments and ministries. Finally, you can announce your budget, which gives you the opportunity to raise money for taxes. At the end of each year of your five-year term an opinion poll tells you how popular your policies were, and will contribute to your ultimate goal of re-election at the end of the game.

1984 is available for the BBC Micro and Spectrum from Incentive Software.

GOLDEN HARVEST

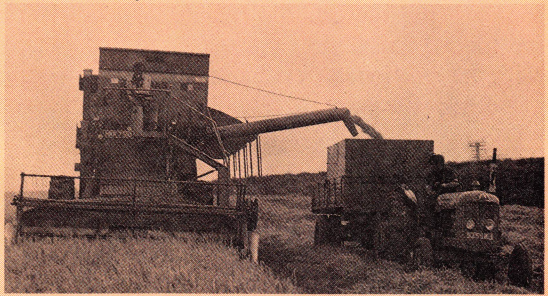
Corn Cropper, available for the Spectrum, Electron or BBC Micro, from Cases Computer simulations, simulates the running of a large wheat farm. You begin the game with £50,000 in cash and the aim is to run the farm so that after five years you have total assets of £250,000.

Initially, you have just 30 acres of land that have to be planted. To do this you need to buy seed, hire a tractor and labour to do the planting and pay for irrigation if there is not enough rain. At the beginning of each month you can see the rainfall and temperature forecasts, and so decide whether to plant or not.

As the months go by, you have to keep an eye on the 'crop status report' — a kind of calendar that tells whether any of the wheat is ready to harvest. If it is, then you will probably need to hire some extra men and a combine harvester to harvest the wheat quickly.

Other hazards can strike: rats might eat some of your seed; if you have not sprayed the crop then insects will attack it; and frost can also cause considerable damage. To increase the yield, fertiliser can be applied, but it has to be applied at the right time.

As the game progresses, you aim to earn a steady income from selling the harvested wheat, so that you can re-invest in more land and seed. It's quite difficult to increase your assets much above about £100,000 — especially as you are not allowed to overdraw at the bank. As a last resort you can always sell off some of your land to raise some more money.



ASSOCIATED PRESS

Armchair Management
These home micro business games use simulation techniques to put you in charge of an airline, a wheat farm, an oil company and the British economy. The degree of realism varies according to the number of control factors that the game supports, and the complexity of the game's factor-interaction models

IAN McKINNELL

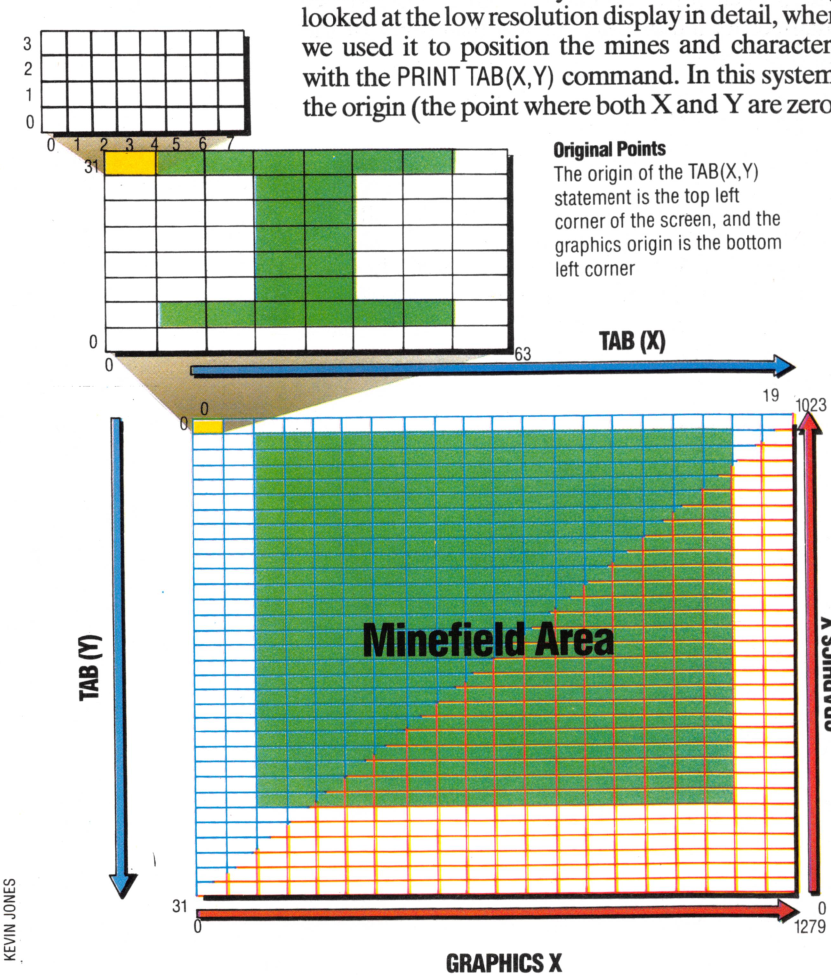
RANGE OF VIEW

This is the second instalment of a project to build up a graphics game for the BBC Models A and B, and the Electron. Here, we look at high resolution line drawing, the BBC's internal timer, and the remaining procedures to set up the game's scenario.

In the first instalment, we defined an area of the BBC mode 5 screen as the 'minefield' on which our game will be played. We defined the shapes of the mines, the detector and the assistant, developed procedures to lay a number of the mines at random in the minefield, and printed the detector and the assistant in their starting positions. To make the display more attractive we can draw a border around the minefield area. The simplest way to do this is to use the high resolution commands MOVE and DRAW.

There is a problem associated with mixing high resolution graphics with low resolution characters on the BBC/Electron: the different displays use different co-ordinate systems. We have already looked at the low resolution display in detail, when we used it to position the mines and characters with the PRINT TAB(X,Y) command. In this system, the origin (the point where both X and Y are zero) is the top left-hand corner. The X values in this system increase from 0 to 19 from left to right, and the Y values increase from 0 to 31 from top to bottom. Mode 2 also uses a 20 by 32 character display, but all other modes show a different number of characters and so use different co-ordinates for each character.

On The Square
Characters are defined in an 8 by 8 pixel matrix. In mode 5, this is actually rectangular, being 64 hi-res dots wide and 32 dots high, so each mode 5 character pixel must measure 8 hi-res dots wide by 4 dots high. PLOTting or DRAWing to any of the dots in a pixel will light the whole pixel—PLOT 7,3 is thus equivalent to PLOT 5,2



Original Points
The origin of the TAB(X,Y) statement is the top left corner of the screen, and the graphics origin is the bottom left corner

KEVIN JONES

The eight display modes of the BBC/Electron offer three different resolutions (640 by 256, 320 by 256 and 160 by 256) and yet they all use the same co-ordinate system. Although this system is rather confusing at first, it is a real help in programming because graphics designed for one mode will fit on the screen in the other modes.

The BBC/Electron co-ordinate system treats the screen as if it has a resolution of 1280 by 1024. This is, of course, a higher resolution than the two machines can produce, apparently to allow for future developments of the BBC! All co-ordinates are given as numbers in the range 0 to 1279 across the screen and 0 to 1023 down the screen. BBC BASIC automatically converts these into the appropriate values for whichever of the three screen resolutions is being used.

Mixing graphics with characters is made a little harder because the graphics origin is at the bottom left of the screen rather than the top left used in the character co-ordinate system.

The 160 by 256 pixel resolution in mode 5 is directly linked to the number of characters that can be shown. Each character is formed from an eight by eight pixel grid. As there is enough room for 20 characters across the screen, there must be $8 \times 20 = 160$ pixels across the screen. Similarly, there must be $32 \times 8 = 256$ pixels down the screen. To relate this to the high resolution co-ordinate system, we must remember that a pixel is the smallest area of light on the screen that can be individually turned on or off. In the high resolution system, there are 1280 different co-ordinates in the X direction. If we divide this by the number of pixels in the X direction we get $1280/160 = 8$. Similarly, in the Y direction, dividing the high resolution co-ordinates by the number of pixels gives $1024/256 = 4$. This means that each pixel can be turned on by plotting to any of several positions in the 1280 by 1024 co-ordinate system. The illustration shows how a range of co-ordinates can be used to turn on (or off) one pixel. Plotting (7,3) would light the same pixel as plotting (0,0) or (5,2) and so on.

We can use this fact to relate character positions to high resolution co-ordinates. On the horizontal axis, if one pixel is equivalent to eight units, then one character width will be equivalent to 64 units. Four units are equivalent to one pixel in the

vertical direction, implying that the height of each character is 32 units. The boundaries of the screen can now be worked out in terms of high resolution co-ordinates for the MOVE and DRAW commands. The Screen Layout illustration shows what these boundaries are.

We can now calculate the co-ordinates of the bottom left corner and the top right corner of the minefield (all other co-ordinates for the border follow from these two points). As we can see from the diagram, the co-ordinates of the bottom left corner of the border are (120,188). The co-ordinates of the top right corner are(1152,992).

The following procedure draws a border around the edge of the defined area. GCOL 0,1 sets the logical colour that will be used for graphics. The first number defines the type of plotting, which will be discussed later, and the second defines the colour. In mode 5, logical colour 1 is normally red. The MOVE commands move the graphics cursor (without drawing) from the origin to the bottom left corner of the border. The DRAW commands that follow it draw straight lines from the current position on the screen to the point specified.

```
2470DEF PROCdraw_border
2480GCOL 0,1
2490MOVE 120,188
2500DRAW 120,992
2510DRAW 1152,992
2520DRAW 1152,188
2530DRAW 120,188
2540ENDPROC
```

THE INTERNAL TIMER

The BBC and Electron have an internal timer that can be accessed easily from BASIC using the reserved variable, TIME. When asked to print the value of TIME, the computer will return a number that corresponds to the time, given in hundredths of a second, since the variable was last set to zero. The procedure 'set-time' prints the word 'Time', its starting value and sets the variable TIME to zero. This procedure is called during the set-up routine and starts the clock for the game.

```
2640DEF PROCset_time
2650PRINTTAB(2,27)"Time      02:00"
2660TIME=0
2670ENDPROC
```

During the main loop of the program, the time displayed on the screen must be updated. To display the time in seconds would be very straightforward; we would simply divide the variable TIME by 100, to convert to seconds, print this value to the screen, and so on. However, it is possible to convert TIME into minutes and seconds by making use of the BBC BASIC commands DIV and MOD. TIME DIV 100 would return the number of seconds as a whole number; (TIME DIV 100) MOD 60 would count the seconds from zero to 59 and then start again from zero. This is because the MOD 60 command gives the value of the remainder after division by 60. So, for example, 63/60 is 1, with a remainder of 3. (63/60) MOD 60 would therefore be 3. The minutes can be similarly isolated and displayed by using (TIME DIV 6000) MOD 60.

This is the procedure that will be used to update the time during the game:

```
2900DEF PROCupdate_time
2910sec$=STR$(((12100-TIME) DIV 100)MOD 60)
2920min$=STR$(((12100-TIME) DIV 6000)MOD 60)
2930REM ** ADD LEADING ZEROS **
2940sec$=LEFT$(zero$,2-LEN(sec$))+sec$
2950min$=LEFT$(zero$,2-LEN(min$))+mir$
2960time$=min$+":"+sec$
2970PRINTTAB(11,27);time$
2980ENDPROC
```

As you can see from this procedure, we have gone a stage further. As well as being divided into minutes and seconds, the time will, in fact, be counted backwards from two minutes to zero. In addition, a short string-handling routine is included to ensure that the displays for the seconds and minutes always have two digits, by adding leading zeros if required.

Two other short procedures are still required to complete the setting up of the game. During the game, the player has four lives; therefore, we need to display, at the bottom of the screen, the number of lives remaining. Initially, this will be three lives, displayed as three of the 'assistant' characters we defined in the last instalment (see page 393). A variable 'count' will be used to determine the number of lives used. Initially, this will be one.

```
2690DEF PROCset_men
2700men$=CHR$(226)+CHR$(226)+CHR$(226)
2710count=1
2720COLOUR 1
2730PRINTTAB(2,30);men$
2740COLOUR 2
2750ENDPROC
```

The final set up procedure initialises the scores and displays them on the screen. The value of 'hiscore$' is not set within this procedure as the procedure is called each time the game restarts. Instead, we shall set its initial value at the beginning of the program only.

```
2770DEF PROCset_score
2780score=0:score$="00000"
2790PRINTTAB(2,28)"Score    00000"
2800PRINTTAB(2,29)"Hi score ";hi_score$
2810ENDPROC
```

Now that all the procedures for the setting up part of the program have been assembled, we can construct a higher-level procedure to call them all. In the last instalment we called all the procedures we had assembled directly from a short main program. You must now delete those lines and add these lines:

```
1880DEF PROCsetup
1890COLOUR 2
1900end_flag=0
1910PROCinitialise_variables
1920PROCdefine_characters
1940PROClay_mines(40)
1950PROCdraw_border
1960PROCset_time
1970PROCset_score
1980PROCset_men
1990PROCposition_chars
2000ENDPROC
```

We are now at the stage where we can write a short main program to call the 'setup' procedure and update the time in a REPEAT...UNTIL loop. Add these lines to your program:

```
10 REM **** CALLING PROGRAM ****
20 hiscore$="00000"
30 PROCsetup
40 REPEAT
50 PROCupdate time
60 UNTIL TIME>12099
70 END
```

VERSATILE PERFORMER

Although it looks like an ordinary electronic typewriter, the Brother EP-44 is an exceptionally versatile machine that can also function as a printer, calculator and communications terminal. To top it all off, the machine costs a mere £250, making it an ideal companion for a home micro.

The Brother EP-44 weighs five pounds and is battery-powered, making it a truly portable machine; a mains transformer is available as an optional extra for a further £20. On the keyboard, in addition to the conventional typewriter keys, are four keys to handle word processing and seven calculator keys. But the most noticeable feature is the 15-character liquid crystal display (LCD). This allows the user to display and edit text before it is printed out.

If the EP-44 is to be used as a conventional typewriter, the Print Mode Selector switch is set to 'Direct Print'. In this mode, text appears in the LCD 'window' and is simultaneously printed out on paper. Tabulation stops, line spacings and margins are set just as they are on a conventional typewriter. However, there is an unusual paper-feed system — instead of manually turning the platen roller, the user must press one of two buttons to move the paper either up or down. The keyboard is not quite up to the standard of an electronic typewriter. but is perfectly adequate.

The EP-44 utilises a thermal print head to 'burn' a series of dots onto the paper and so form the desired character. Most thermal printers use a nine by seven matrix of dots to make up the characters, but the EP-44's 24 by 18 matrix gives higher quality print. Another printer in the Brother range, the EP-22, uses fewer dots and produces noticeably poorer print.

A major disadvantage of the thermal print system is the slow speed of the printout — in this case under 16 characters per second. When the Brother is used as a typewriter this is not a drawback, but when the machine is used as a computer printer the slow speed is very noticeable — a good dot matrix printer will produce text at 10 times the Brother's speed. The other problem with this system is the special thermal paper that is required; this is expensive and has a very glossy appearance. The thermal ribbon may be used with normal paper, but the print quality will suffer.

If the Print Mode Selector switch is moved to 'Correction Print', the LCD window shows its worth. In this mode, text appears in the window

The Keyboard
The 51 printing keys are slightly sculptured, with a soft action, in a non-standard QWERTY layout. There are two character shifts with lock, a symbol shift and a function shift. The character keys are multi-function, giving a range of foreign characters and accents. 21 function keys control print, calculator and memory modes, margin setting, tabs, and paper movement

IAN McKINNELL

but there is a delay before it is printed out. In fact, text printed on paper is 15 characters behind the letter currently being typed. This feels strange at first, but it allows you to alter any of the last 15 characters before they are printed — useful if your typing is not very accurate, and certainly a great improvement on correction fluids. Editing is done on the LCD display by using the two cursor keys to select the character to be altered. There is also a 'Line-By-Line' mode, in which the EP-44 waits until the Return key has been pressed before printing an entire line of text. This gives you the chance to edit a whole line instead of just the last 15 characters typed. In this mode, the LCD display acts as a window that is moved along the line of text by the cursor keys.

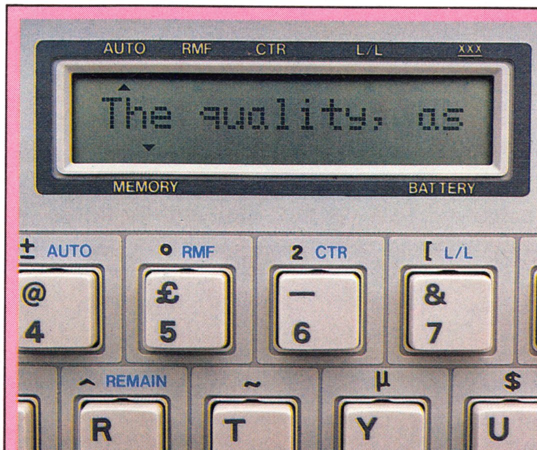
Word processor users become accustomed to the fact that text is formatted automatically and soon lose the habit of pressing the Return key at the end of a line of text. The EP-44 has an 'Auto Return' mode, in which a new line is automatically started if the space bar is pressed within six character positions of the end of a line.

The EP-44's built-in calculator allows simple calculations to be performed on the LCD display without disrupting printing. For example, if you are typing out a sales invoice and need to work out VAT payments or trade discounts, you can use the calculator to work out the relevant figures and then continue printing out the invoice details.

All the features mentioned here can also be found on many modern electronic typewriters. But the EP-44 may also be used as a simple word processor. This feature utilises the 3.5 Kbytes of internal memory that is used by the Brother to store text. This memory may be small when compared to that available on most home computers, but it is sufficient to produce a three-page letter. The commands necessary for word processing are accessed by using the normal typewriter keys in conjunction with a special blue 'Code' key. Thus, to enter text into memory, all that is needed is to press the Code key and the letter 'N' (for 'New text'). Text can be entered in both Direct and Correction Print modes.

Once text is entered into memory, the LCD display and the cursor keys are used to edit it. The cursor keys enable the LCD window to be moved up, down, left and right, so the entire text can be viewed and characters inserted, deleted or altered at will. The only restriction is the fact that words may not be moved down to the line below if the line being edited is too long — which means you may not re-format a document.

The great advantage of entering text in this manner is that it may be altered without the necessity of retyping the whole document. For example, you may need to send the same letter to several different people; to do this, you simply edit the text in memory to insert the relevant name and address, and then press the Code key together with the letter 'P' (for 'Print text'). Your modified letter will then be printed out. Text remains in the EP-44's memory even after the power is turned off, allowing you to enter text and edit it later.



LCD—OK?

The LCD screen of the EP-44 displays 15 characters, five print mode indicators and a battery charge warning. Text can be edited on the screen, thus allowing simple word processing. The display density of the characters can be adjusted to cope with ambient light conditions

IAN McKINNELL

Removal of a small cover on the side of the EP-44 exposes an RS232 serial socket. A switch marked 'Terminal' turns the Brother into a computer printer, although some experimentation may be needed to set the correct baud rate, word length, and so on, for your particular machine. Setting up the EP-44 is extremely easy: the LCD display cycles through the various baud rates and other details, and you simply press the Mode switch when the details are correct for your machine. These settings are maintained even after power has been switched off.

Unfortunately, the Brother cannot handle the usual fanfold paper, and single sheets must be fed in by hand. But the quality of print is much better than on a dot matrix printer, so the EP-44 is ideal for the production of letters and short documents.

The fact that the switch is labelled 'Terminal', not 'Printer', shows that the EP-44 can send data as well as receive it. This means that the Brother may be connected to a modem, allowing the user to communicate by way of the telephone with other home computer users, or larger machines that use suitable communication standards.

All in all, the Brother EP-44 is notable for its versatility and the fact that so many different functions have been packed into such a small machine. At a cost of £250 it represents excellent value and should be considered by any home computer owner who is looking for a computer printer that can also be used as a typewriter.

Print-Out

Any smooth-finish typing
paper can be used with the
ribbon in place, but the
paper-feed cannot accept
carbons. The print
quality is good on both
types of paper.
Text can be
Centred & Underlined
foreign (Æø¿¢µ¶), and
sub- or super-scripted
(X^2=N_2+T_0).

D

DATA PROCESSING

Before the microcomputer was developed, all computing was known as *data processing*. At that time, the high cost of mainframe and minicomputers limited their use to very large firms and public bodies. The hardware needed special air-conditioned environments, and required trained staff to operate and maintain it. The computers were invariably controlled by a Data Processing (or DP) department, and no one could gain access to the machine without the approval of the Data Processing Manager.

The microcomputer revolution meant an increase in accessibility. The micro, freed from the constraints imposed upon mainframes and minicomputers, could sit on an executive's desk, where it would be constantly available to perform whatever application was required. Data Processing departments still exist in large companies, and perform a wide variety of tasks. These applications include financial modelling, stock control, cost and management accounting, production planning, payroll and personnel records.

DEBUGGING

In its broadest sense, *debugging* is simply the correction of errors — whether these errors are mistakes in a computer program or malfunctions in the hardware. In particular, though, the term is used to refer to the systematic testing and correction of a computer program.

The person credited with coining the term is Captain Grace Hopper, who worked on the electromechanical Harvard Mark II computer in 1945. The cause of a particularly elusive error in a program was eventually traced to a moth that had been hammered to death in one of the machine's relay switches. When asked by a superior why progress was slow, Captain Hopper explained that she was 'de-bugging the machine'. From this derives the word *bug*, which is simply an error in a program's operation.

DECISION TABLE

A computer's usefulness is derived from its ability to act on conditional statements. The simplest of these is the BASIC `IF...THEN` statement, in which the computer follows a set course of action if a particular condition is met. All forms of computer decisions can ultimately be reduced to this form.

A *decision table* simply indicates the actions to be taken under various conditions. It is set up in two columns — one for the *conditions* and the other for the *actions* to be taken. The computer then works through the first column until it finds the entry that matches the required condition and takes the action indicated by the second column.

Implementing a decision table in BASIC by any means other than a collection of `IF . . . THEN` statements would be a very involved process, so machine code is generally used. The `ON...GOTO` structure is, however, a simple form of decision table, in which the list of conditions consists of a list of numbers, and the possible outcomes are all `GOTO` statements.

DECISION TREE

Like a decision table, a *decision tree* is a collection of conditions and specified actions. The difference between the two is that in a decision tree the computer does not need to search the entire list. Instead, it searches an initial section of the list and the resulting action statement then indicates the next group of conditions to be searched. The effect is like a tree — the program starts at the trunk, and each set of conditions that is met represents a split into several branches.

DECLARATION STATEMENT

The BASIC language requires all array variables to be `DIM`ensioned before use. Some other programming languages insist on *all* variables being 'reserved' in this way, with the user having to specify whether they are integer, real, string or double-precision variables. The program lines that reserve variables (usually at the start of a program) are called *declaration statements*.

There is no fundamental reason why arrays must be declared, except for the fact that this makes the system more efficient. When a `DIM` statement is encountered, the BASIC interpreter sets aside an area of RAM proportional to the size of the array. Ordinary variables are then stored in RAM adjacent to the array variables. If a `DIM` statement was not used, each time the user addressed the array using a higher index than before, the operating system would need to move all the other variables in memory to make sufficient space.

It is good programming practice to simulate the declaration of all variables at the start of a program, in the order of frequency of use. Thus, if your program features many loops that use `I` as an index, the statement `LET I = 0` in the first line of the program will make it run faster by ensuring that `I` is placed in the first position in the table of variables.

TRACE ELEMENTS

Although many home computers have excellent graphics facilities, transferring a picture or design from paper to the computer's display can be a time-consuming and extremely difficult task. A far better option is to choose one of the many digital tracers that are marketed for use with microcomputers.

A digital tracer is a simple piece of equipment that allows you to trace over a drawing, photograph or design, and at the same time transfer the image to the computer's display. The ease with which this may be done is largely dependent on the software that accompanies the equipment. Here, we look at four digital tracers — three for the BBC Micro and one for the Sinclair Spectrum.

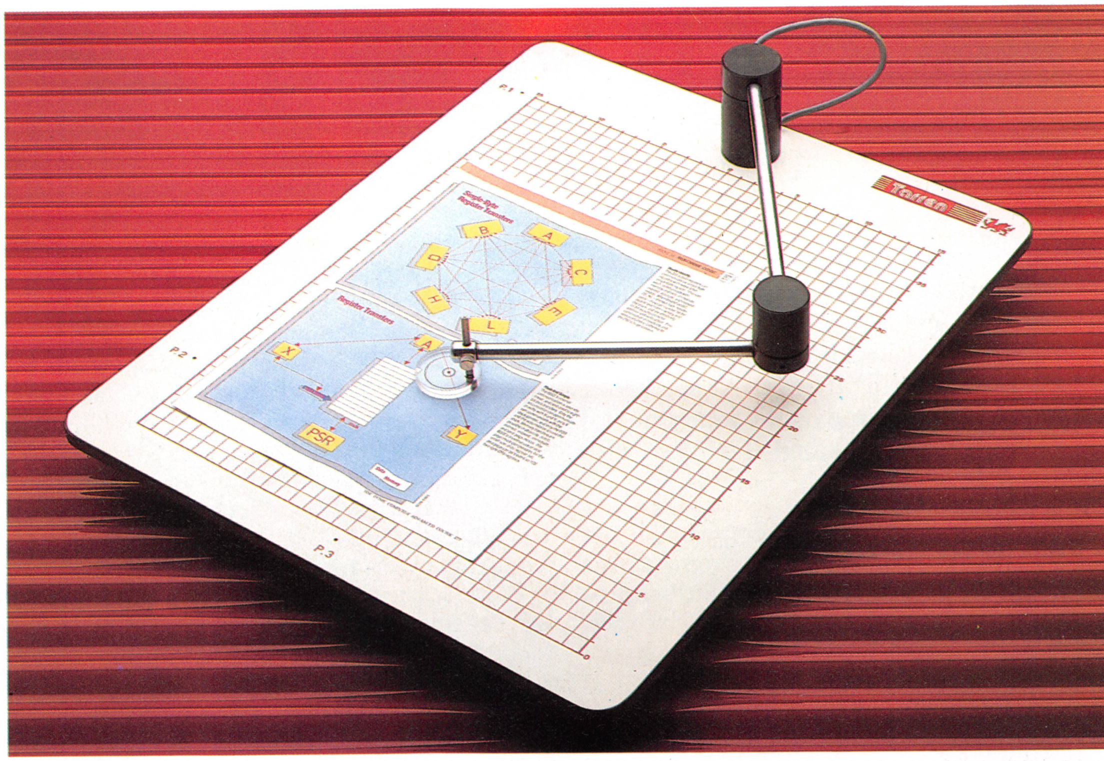
All of the tracers work in a similar way. A pointer is fixed at the end of a double-jointed arm, which sends electrical signals to the computer. These signals vary in strength, depending on the position of the pointer. They are converted into digital format by the computer and used to plot a point in the correct place on the screen. All the tracers are supplied with software to carry out this task, offering various options such as drawing lines in different colours. The software for the BBC models allows different display modes to be selected, trading off resolution against the number of colours available within the constraints of limited memory space.

The Robot Plotter, from Robot Computer Development, is the most impressive looking of the four tested. This model has a perspex base on which is inscribed a grid showing pixel positions. The tracer arm is anchored at one end of the base. The picture to be traced may be placed underneath the grid and viewed through the clear base. The arm on this model is extremely sturdy, being constructed from thick metal and plastic. The pointer is a pencil-like stub that juts down from the arm to the base plate. Unfortunately, with this system it is not easy to see the picture as it is being traced.

The Robot Plotter costs £69 and is sold with a single cassette containing software to run on the BBC Micro. In addition to the tracing routines, the cassette contains several circle, rectangle and line drawing routines that are used in conjunction with the arm.

The tracing program stores all images as a series of lines; thus, a map tracing would be held in memory as a sequence of short lines. This makes it easy to remove unwanted lines without affecting other nearby lines. However, a complex image requires a lot of memory and it is possible to use up all the limited spare memory of the BBC machine. Because the display is stored as a sequence of lines, it is relatively easy to transfer images created with

Robot Plotter
Artwork for tracing can be placed under the perspex work table of the Robot Plotter; the software provided includes circle and rectangle routines



Digigraph
Solidly built from metal and plastic with a wooden tracing table, the Digigraph package includes several practice worksheets

the tracer to other programs.

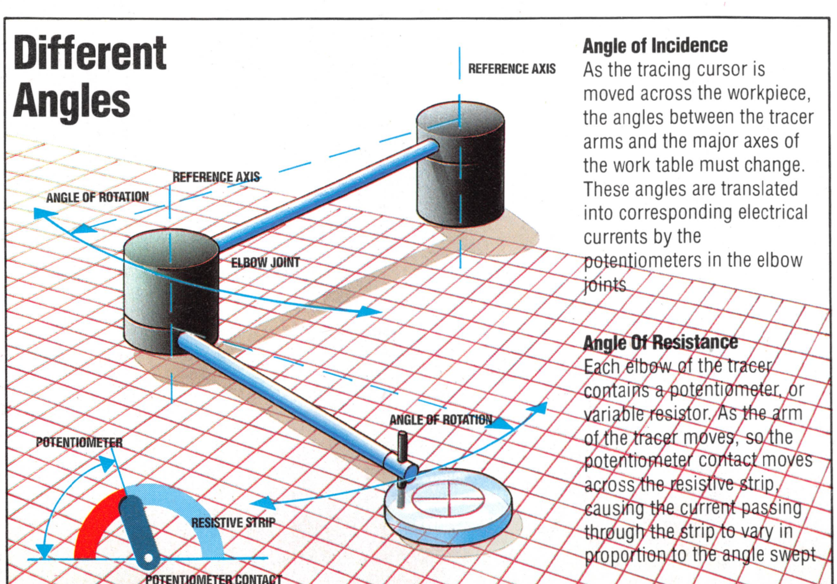
The Digigraph tracer is also solidly built. The base plate consists of a large wooden board with a red grid painted on the surface. The arm is made up of aluminium tubing, with a perspex disc at the pointer end. The picture to be traced is placed on the board. This system is the easiest to use of the four tested here; the arm moves smoothly and the picture is clearly visible through the perspex disc. The supplied software is less sophisticated than that sold with the Robot Plotter, but has similar facilities. The tracer movements are not stored as separate commands by the computer, but pictures are drawn straight onto the screen and saving and loading pictures is achieved by saving and loading the screen memory. This means that a picture cannot be easily edited, but it has the advantage of using no more memory to draw a complex picture than to draw a simple one — which is extremely useful on the memory-starved BBC Micro. The Digigraph system also includes several 'worksheets', which the user is invited to use to practise copying. Together with the excellent manuals, these allow you to master the system with the minimum of effort. The Digigraph costs £100 with cassette software or £105 with the software on disk.

The RD Labs tracer is marketed in two forms — one for the BBC and the other for the Spectrum. Both versions are virtually 'do-it-yourself' kits: the tracer arm only is supplied, along with a peel-off sticky pad that allows you to glue the arm to a suitable base. The arm is made of plastic and is very flexible, but operates reasonably accurately. The pointer is a plastic cross-hair arrangement that is somewhat cumbersome in use.

The BBC version is priced at £70. For this, you get the arm itself, a manual and a software cassette. The pictures are stored in a similar way to those produced on the Robot Plotter, with the result that memory shortage is again a problem. The software is fairly complex, containing routines for drawing circles, rectangles and lines and an animation facility. This uses the BBC's ability to redefine the colour palette to animate short simple sequences. A demonstration cassette shows off all the features of this tracer.

The Spectrum version is the least expensive of the four at £56. The package includes an add-on analogue-to-digital converter (the BBC Micro already has this built in). In most other respects, the Spectrum model is very similar to the BBC version. Continuous drawing is possible, but is difficult to do satisfactorily because of the slow response of the software, and a smooth curve

Different Angles



Angle of Incidence
As the tracing cursor is moved across the workpiece, the angles between the tracer arms and the major axes of the work table must change. These angles are translated into corresponding electrical currents by the potentiometers in the elbow joints

Angle Of Resistance
Each elbow of the tracer contains a potentiometer, or variable resistor. As the arm of the tracer moves, so the potentiometer contact moves across the resistive strip, causing the current passing through the strip to vary in proportion to the angle swept

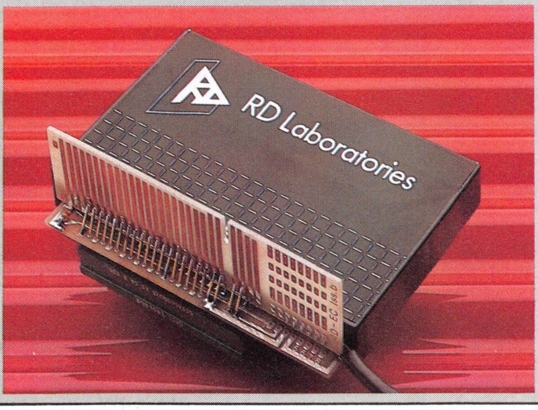
RD Digital Tracer
This version for the BBC Micro is simply constructed in lightweight plastic; a strip of non-removable adhesive tape is provided for table-top fixing

tends to be displayed on the screen as a series of straight lines. But all the other facilities (for drawing circles, rectangles, etc.) are supplied. The software does not produce an error report — unlike normal Spectrum programs — if these commands are used to draw lines that extend beyond the boundaries of the screen.

It is possible to write programs that read positions directly from any of these tracers, which is useful for certain special purposes. However, this is difficult to achieve on the Spectrum, as this computer, unlike the BBC, has no joystick port and lacks the relevant commands in its BASIC.

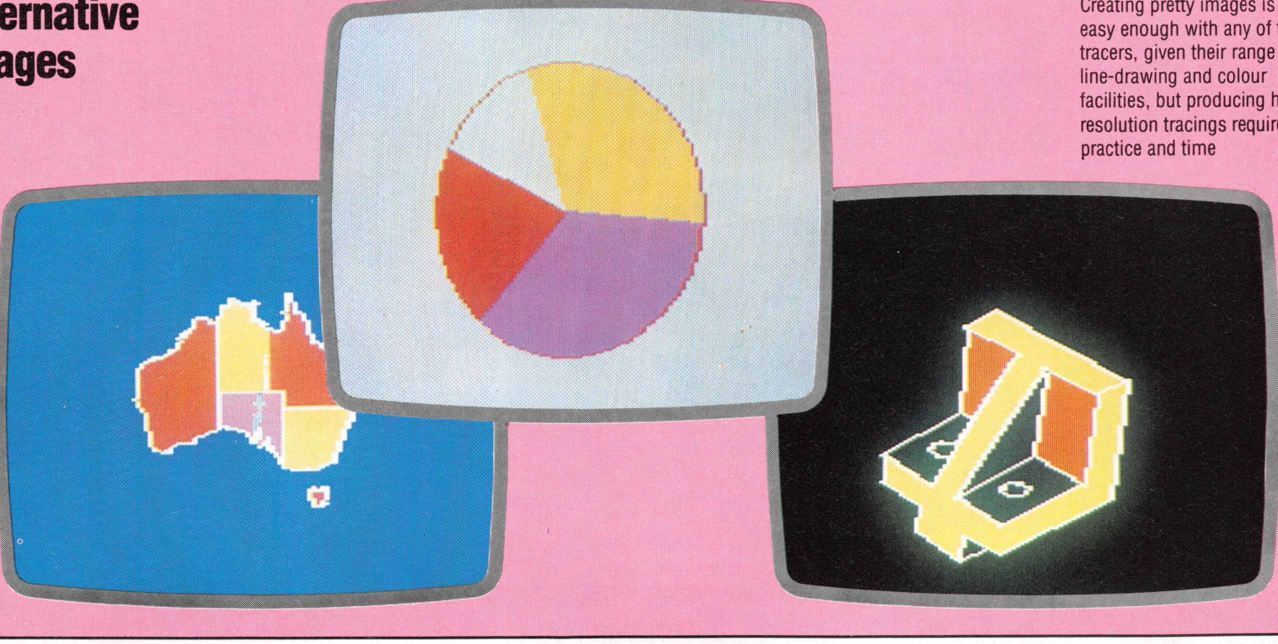
For Spectrum owners, the choice of tracer is limited to the RD Labs model, but this works well enough. The BBC user has a wider choice: the RD Labs tracer is the cheapest of the three, but its construction is somewhat flimsy; the Robot Plotter is solidly built and is supported by some clever software, although it is not particularly easy to use; the Digigraph has less sophisticated software but is by far the easiest to use.

RD Spectrum Interface



The RD Tracer for the Spectrum is exactly like the BBC version shown in this article, but is connected via its own interface

Alternative Images



Creating pretty images is easy enough with any of the tracers, given their range of line-drawing and colour facilities, but producing high-resolution tracings requires practice and time

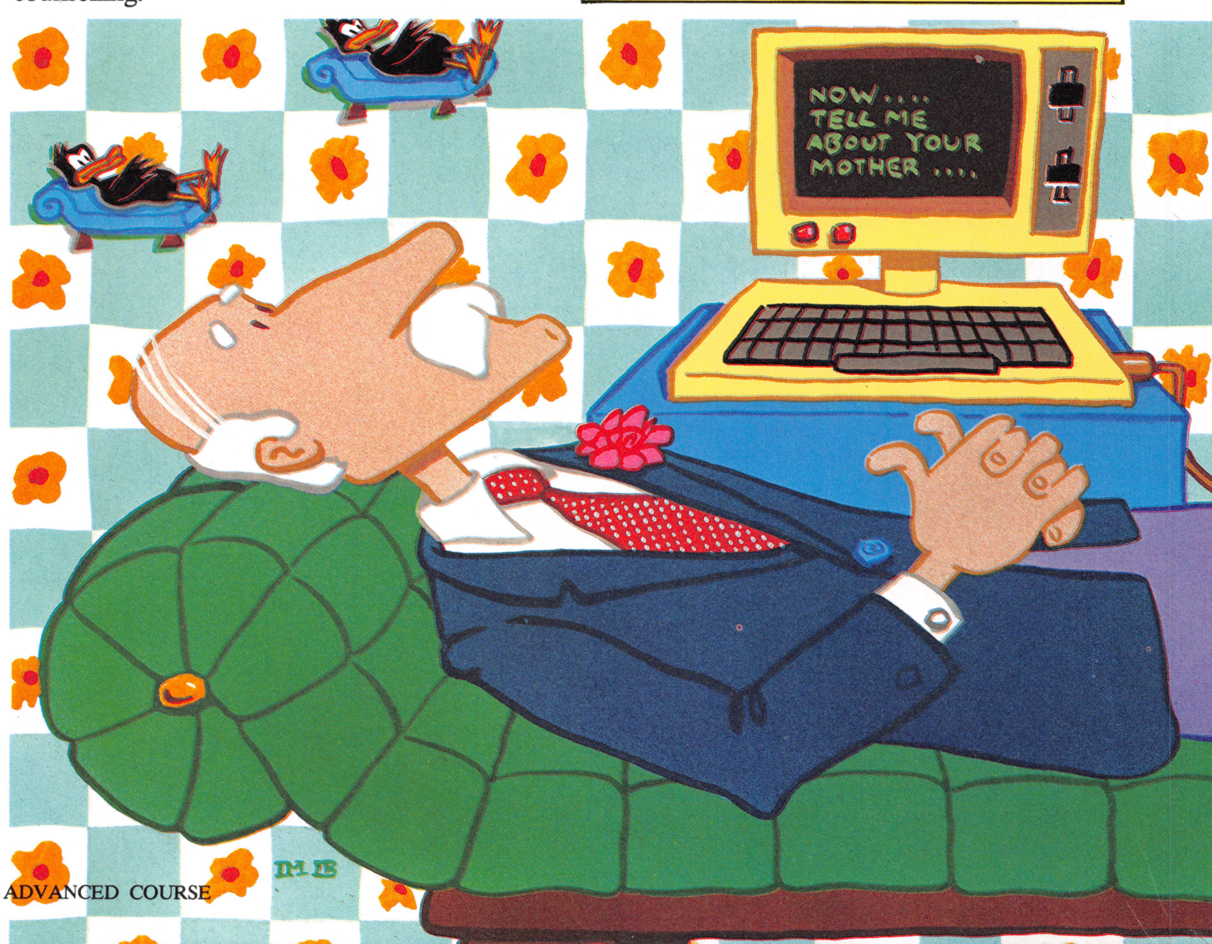
PRIVATE CONSULTATION

The 'fifth generation' of computers will not require programming as we know it, but will 'understand' 'natural languages' — their operating systems will accept and carry out commands expressed in the user's own language. Here we present a simple simulation that illustrates the principles of such a system.

One of the pioneers of computing, Alan Turing, proposed a famous test of artificial intelligence: he said that if a machine could converse with a human, and the human couldn't tell that he was talking to a machine, then the machine could be said to be 'intelligent'. So far, no machine has truly passed that test, but several programs approach it by deflecting the human's interest in the conversation from the machine's responses, so that the human does all the talking. This may sound far-fetched, but it's surprisingly effective. If you think about an enjoyable conversation you've had, you'll probably find that you did most of the talking, while the other person simply supplied conversational 'noises' or prompts. This is the principle of many types of psychotherapy and counselling.

Program

```
100 GOSUB 2000:   REM INIT
200 FOR L=1 TO 2*LT STEP 2
300 PRINT O$:INPUT I$
400 GOSUB 3000:   REM ANALYSIS
500 NEXT L
1000 PRINT  TAB(5); "-----THE SESSION'S
OVER-----"
1100 PRINT  TAB(4); "--PRESS ANY KEY TO
CONTINUE---"
1200 A$=INKEY$:IF A$="" THEN GOTO 1200
1300 GOSUB 5000:  REM REPORT
1900 END
1999 REM************************
2000 REM**   INITIALISE  S/R  **
2001 REM************************
2050 LT=10:AN=10:T9=500:EX=2*LT
2100 DIM R$(AN):DIM H$(2*LT)
2200 DATA "YES..","TELL ME MORE..", "GO
ON..","AND..","SO.."
2250 DATA"IS THAT IMPORTANT..","WHY DOES
 THAT MATTER.."
2300 DATA "PLEASE EXPLAIN THAT..", "WHY
DO YOU SAY THAT.."
2350 DATA "HOW DOES THAT AFFECT YOU.."
2500 FOR K=1 TO AN:READ R$(K):NEXT K
2600 CLS:O$="HI - WHAT'S THE TROUBLE.."
2950 RETURN
2999 REM************************
3000 REM**   ANALYSIS    S/R  **
3001 REM************************
3100 IF I$="GOODBYE" THEN EX=L-1:L=2*LT:
RETURN
3200 H$(L)=O$:H$(L+1)=I$
3300 R9 = INT(AN*RND+1):IF R9=RO  THEN
 GOTO 3300
3400 O$=R$(R9):RO=R9
3950 RETURN
4999 REM************************
5000 REM**   REPORT      S/R  **
5001 REM************************
5050 CLS:PRINT TAB(10);"**REPORT**"
5100 FOR K=1 TO EX STEP 2
5150 PRINT TAB(5);H$(K)
5200 PRINT H$(K+1)
5300 FOR D=1 TO T9:NEXT D
5400 NEXT K
5900 RETURN
```



MICHAEL BROWNLOW

The program listed here simulates the behaviour of a completely non-directive therapist, in other words a therapist who does not *direct* the conversation, but keeps it going with responses such as 'TELL ME MORE . . ' and 'WHY IS THAT IMPORTANT . . ?'. It keeps a record of the conversation, and replays it after a given number of exchanges, or as soon as you type 'GOODBYE'.

The program can be developed in the direction of pseudo-intelligence by making it analyse the user's input and choosing an appropriate reply. It already does this in line 3100 by testing for the word "GOODBYE". We might extend this analysis by testing for the words "YES" and "NO", and making more specific prompts in answer — something as simple as "WHY?" or "WHY NOT?" would do very well. Next, we could check whether the user is repeating a response, and, if so, reply accordingly or end the session. We might test whether a response ends in a query or an exclamation mark, and reply "WHY DO YOU ASK THAT . . ?" or "WHY ARE YOU GETTING EXCITED . . ?". These are all quite effective strategies, and do not require a lot of processing time, or any attempt at understanding the sense of the user's response.

We might now set up a table of keywords, and search the response for these words, making a specific response from another table if we find them. The choice of keywords and answers depends on the kind of conversation you expect to have; the way that your choice affects the user's responses can give some fascinating insights into your own and your friends' subconscious minds. The drawback of this method —and all methods that must search the text — is the time it takes to analyse even a short sentence. The speed of BASIC is the limiting factor here, and any serious analysis requires a machine code program. We can, however, tolerate delays for the sake of investigation. Even these simple methods can give surprising results, and illuminate the problems that the fifth generation machines have to solve.

If we intend to analyse the user's words, then there are many approaches to take: the most interesting is probably *syntactic analysis,* the reduction of a sentence to its component parts of speech, such as pronoun, verb or noun. This requires a body of syntactical and grammatical rules, and tables of pronouns, prepositions, conjunctions, word transformations and so on, and is not a simple matter. It would be easy, however, to choose the longest word in the response, and ask the user to explain his feelings about it — the longest word is likely to be the most important in a simple sentence. We could improve that, perhaps, by choosing at random among those words longer than, say, five letters, or, use the word following "I", or "MY" or "YOU" or "YOUR".

Choosing among, and refining, such methods is a fascinating programming exercise. You get a very clear view of the immense complexity of language and its analysis, and may become dissatisfied with BASIC as an analytic tool. This could lead you to other programming languages such as LISP and PROLOG, which are structured far more subtly, precisely because of the need to cope with the complexity of language and thought. In addition to this, of course, you also get the chance to talk as long as you like to the perfect conversationalist — someone who listens to you!

```
3420 IF LEFT$(I$,3)="YES" THEN O$="WHAT
MAKES YOU SO SURE..":RETURN
3450 IF LEFT$(I$,2)= "NO" THEN  O$="WHY
NOT..":RETURN
3500 Z$=RIGHT$(I$,1)
3520 IF Z$="?" THEN O$="WHY ASK ME.."
3550 IF Z$= "!" THEN  O$="WHY DOES THAT
UPSET YOU.."
3600 IF R9<AN/4 THEN GOSUB 4000
3950 RETURN
3999 REM************************
4000 REM**  LONGEST WORD S/R   **
4001 REM************************
4050 W=1:H=1:WL=1:S=1
4100 I$=I$+" ":L9=LEN(I$)
4120 FOR C=1 TO L9:FOR P=C TO L9
4150 Z$=MID$(I$,P,1)
4170 IF Z$=" "THEN W=P-C:H=C:C=P:P=L9
4200 NEXT P
4220 IF W>WL THEN WL=W:S=H
4250 NEXT C
4270 O$ = "WHAT DOES " + MID$(I$,S,WL) +
 " MEAN TO YOU.."
4450 RETURN
```

Add these lines to the first program for YES/NO and longest word detection

```
**REPORT**

   HELLO - WHAT'S THE TROUBLE..
THE FALCON CANNOT HEAR THE FALCONER
   TELL ME MORE..
THINGS FALL APART - THE CENTRE
CANNOT HOLD
   GO ON..
MERE ANARCHY IS LOOSED UPON THE
WORLD
   PLEASE EXPLAIN THAT..
THE BEST LACK ALL CONVICTION
   IS THAT IMPORTANT..
THE WORST ARE FULL OF PASSIONATE
INTENSITY
   YES..
SURELY SOME REVELATION IS AT HAND
   PLEASE EXPLAIN THAT..
SURELY THE SECOND COMING IS AT HAND
   WHY IS THAT IMPORTANT..
AND WHAT ROUGH BEAST - ITS HOUR
COME ROUND AT LAST
   GO ON..
SLOUCHES TOWARDS BETHLEHEM TO BE
BORN
```

Stimulating Thoughts

We used some lines from W.B. Yeats's poem 'The Second Coming', to illustrate the program's ability to stimulate thought in the user

Basic Flavours

This program is written in Microsoft BASIC. Spectrum users must insert LET in all assignment statements, and adjust the TAB values.

Spectrum

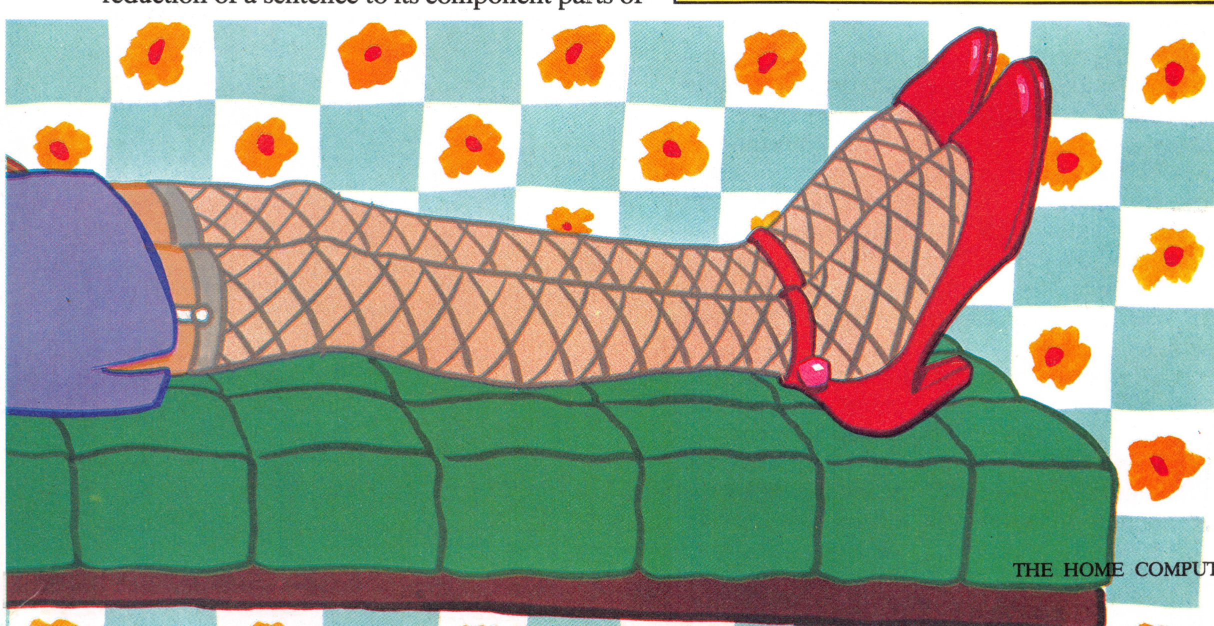
Replace DIM R$(AN):DIM H$(2*LT) by DIM R$(AN,30):DIM H$(2*LT,100)
Replace LEFT$(I$,3) and LEFT$(I$,2) by I$(TO 3) and I$ (TO 2)
Replace RIGHT$(I$,1) by I$(LEN(I$))
Replace MID$(I$,P,1) by I$(P)
Replace MID$(I$,S,WL) by I$(S TO S+WL-1)

Commodore Vic-20 and 64

Replace CLS by PRINT CHR$(147)
Replace INT(AN*RND+1) by INT(RND(1)*AN+1)
Replace A$=INKEY$ by GET A$

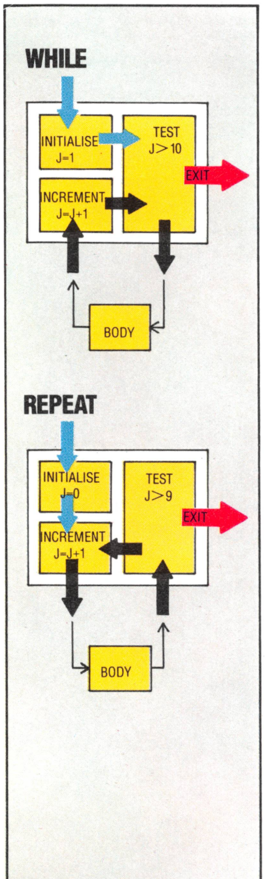
BBC Micro

Replace A$=INKEY$ by A$=INKEY$(0)
Replace INT (AN*RND+1) by RND (AN)



LOOP LINES

Flowcharts are an important technique in program design, but the commonplace notation is not always sufficiently precise, especially when dealing with loop structures. We consider various ways of classifying loops, and introduce a new flowchart symbol — the iteration box.



Iteration Box

Iteration, or looping, is one of the essential structures of any programming language. Earlier in the course, we said that a loop was used in an algorithm whenever a decision switched the flow of control onto a path that brought it back, eventually, to the same initial decision. This adequately describes the structure: the repeated performance of a body of code. However, it doesn't define it in all its many forms. Since loops are such an essential primitive structure, comprising probably 60 per cent of all processor activity, it is extremely useful to look at them in more detail. We will pay particular attention to their effect on general program/algorithm structure, and to the various ways in which they are constructed and classified.

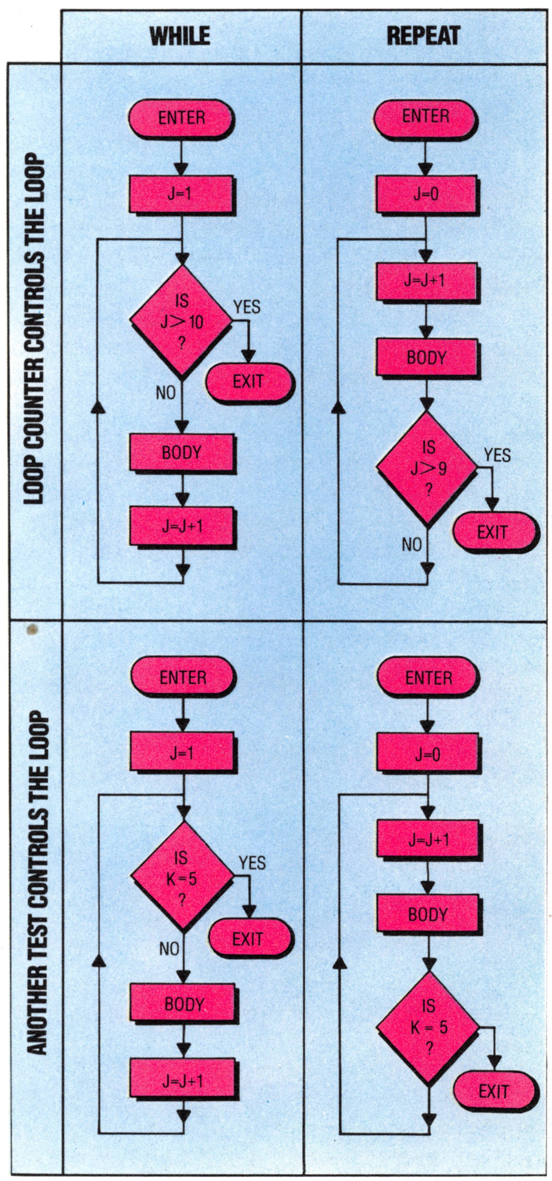
Loops are often divided into two classes, depending on their similarity to the two high-level language loop structures, REPEAT...UNTIL and WHILE...ENDWHILE; both types are used in PASCAL, and the REPEAT loop is implemented in BBC and Oric BASIC. The two types differ in their positioning of the loop exit test: in a REPEAT loop the test comes at the end of the loop body, whereas in a WHILE loop it comes at the start. This means that the body of a REPEAT loop, once entered, will always be executed at least once, whereas that of a WHILE loop need not be. This first difference can be seen quite clearly in the flowchart diagram.

Another way of classifying loops is according to whether the variable that acts as the loop counter is used in the loop exit test, or whether some other test condition controls the loop exit. This is less clearly seen in a flowchart of the conventional linear kind. In fact, the kind of 'commonsense' flowchart that we have become familiar with portrays loops in an unhelpful way. In these flowcharts, a loop looks exactly the same as a simple branch, and it is often necessary to examine the algorithm in detail to distinguish between them.

A clearer notation, called the 'iteration box', exists, which clearly marks the start of a loop, and eliminates the loop/branch confusion. It consists of three linked boxes: the first box shows the initialisation of the counter, the second shows the incrementing of the counter, and the third contains the loop exit test. REPEAT and WHILE loops can both be shown in this form, but differ in the flow of control through the boxes, while the test box indicates whether the loop is counter-controlled or not. These points can be clearly seen in the diagram.

In a REPEAT loop the flow of control is 'initialise-body-test-body-test', whereas a WHILE loop goes 'initialise-test-body-test-body'. This can be seen in the way that the control lines leave and re-enter the iteration box, with the body of the loop 'dangling' from them.

Loop Classification



POT BLACK

With the introduction of colour television in the late 1960s, the game of snooker became a mass-appeal spectator sport. At first sight it may appear an unlikely candidate for computer simulation, but a simplified version of the game is now available for three popular home computers.

To anyone who has ever played it, the idea of computerising the game of snooker is anathema. To a professional player it is sacrilege. How, they ask, could every subtle curve of a ball's flight, or every variation in the table's character, be quantified and programmed? And certainly not within the confines of a 16 Kbyte Spectrum! However, the game of snooker can be thought of as a fairly simple application of the principle of conservation of momentum, and the mathematics of this are easily handled by a computer.

When you begin playing Visions' Snooker, you are presented with a plan view of the table, with the balls represented by different coloured circles. You aim the white ball by moving a cross on the screen to a target position, thus defining the path in which the ball will travel. The white ball is then fired towards its target with a strength that is determined by the length of time that the key is held down.

After playing a few shots, a seasoned player would be aware of two significant problems. The first is that the computer — whether Spectrum, BBC or Commodore — cannot calculate the angles of play or the friction drag well enough to give convincing results on a television screen. This means that many shots give quite unpredictable results (although this can be true in a real game, too). Some shots you may want to play are impossible on the computer. To compensate for this, the pockets are proportionately larger than they should be.

The other problem, which in practice is more profound, is that you are looking at a plan view of the table. This means that when you come to play a shot, you lack the advantage of sighting along the cue. And because a television screen is slightly curved, judging angles is made doubly difficult. Your initial attempts at this game are, therefore, likely to be highly frustrating.

Nevertheless, once you've accepted that the game bears little relation to real snooker, and after you've begun to master a few of its eccentricities, it becomes a quite fascinating game of strategy in its own right. You begin to notice some subtle features of the programming — in particular, it is possible to add spin to the ball, although the results of this are once again unpredictable.

The three versions of the game differ in ways that reflect the strengths and weaknesses of the machines that they're played on. The game uses surprisingly little memory and therefore loads quickly on all three machines — even on the Commodore, which is notorious for its slow loading speed. Consequently, the quality of each version reflects the graphics capabilities of each machine, and not its memory capacity.

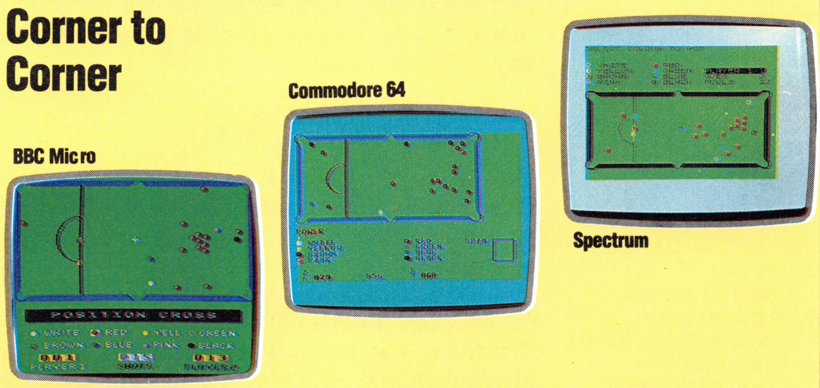
The Spectrum is weakest in this respect. The table is small, and the balls are not all displayed in their true colours. The Commodore version is better, with a larger table, realistic colours and balls that are more accurately scaled. The score is more strikingly illustrated on this version, and there is a demonstration option at the beginning of the game — although you have to complete a game before you are handed back control. The BBC version is the best of the bunch, with a table that covers virtually the full width of the screen, and is strengthened by a much finer control of the cue.

The game has a facility for one or two players, although the one-player option is really only for practice. As so many computer games are extremely anti-social, the two-player game scores points in this respect. Unfortunately, Visions have not seen fit to provide smoking jackets with the software. The over-riding impression given by the game is that it is worthy, but you suspect that a lot of its potential — especially with so much unused memory — has been largely undeveloped.

Snooker: For Commodore 64, £8.95
For 16K Spectrum, £8.95
For BBC Micro, £8.95
Publishers: Visions (Software Factory) Ltd, 1 Felgate Mews, Studland Street, London W6 9JT
Author: Tim Bell (BBC Translation by Andy Williams)
Joysticks: Optional
Format: Cassette

The essence of snooker is accurate aiming and precise weighting of the shots. In Visions' snooker game, an aiming cursor must be positioned on the table while strength of shot is controlled by holding and releasing the fire button

Corner to Corner



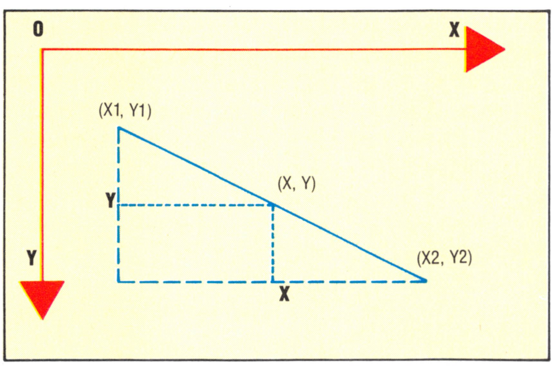
IAN McKINNELL

MAKING THE GRADE

Earlier in the machine code course we developed a program called Plotsub for the Commodore 64, which allowed us to plot high resolution points (see page 337). Here, we take a look at another routine for the same machine that makes use of Plotsub to draw lines of various gradients.

The most obvious way to draw a line from one point to another on the screen is to calculate the gradient of the line using the co-ordinates of the two endpoints. Thus, starting at one end of the line, we could plot each successive point by incrementing the X co-ordinate, and calculating the corresponding value of the Y co-ordinate using the value of the gradient. Each point of the line would be plotted until the other endpoint was reached. Let's look at the mathematics of this technique in more detail.

If we call the endpoints of the line (X1,Y1) and (X2,Y2), then the gradient, G, will be equal to (Y2-Y1)/(X2-X1). The diagram shows a line with one of the points on it labelled (X,Y):



The gradient between the starting point and (X,Y) is given by calculating (Y-Y1)/(X-X1). This gradient is, of course, equal to the gradient for the whole line, G. By equating these two elements, and rearranging their terms, we can arrive at a mathematical expression for the Y co-ordinate of any point on the line:

$$(Y-Y1)/(X-X1) = G$$
$$(Y-Y1) = G(X-X1)$$
$$Y = Y1 + G(X-X1)$$

If the step between X and X1 is only one unit (i.e. if we are incrementing the X value by one), then this reduces to:

$$Y = Y1 + G$$

Therefore, starting at the endpoint (X1,Y1), and incrementing X, each succeeding value of Y can be calculated by repeatedly adding the value of the line's gradient.

You may like to try implementing this technique in BASIC using Plotsub to plot each point. However, there are difficulties with this method:

- If the value of X2 is less than that of X1, then we will need to decrement X rather than increment it.
- If the gradient is greater than one, then the line will not be 'continuous'. This is because when G is greater than one, Y will increase by more than one for each increment of X.
- Negative gradients will cause difficulties when implemented in machine code. Two's complement arithmetic could be used to represent negative changes in X and Y, but as the Y co-ordinates can range from 0 to 199 and X values range from 0 to 319, then *two byte* two's complement arithmetic would be needed.
- Vertical lines cannot be drawn, as calculating their gradients involves division by zero.

MAKING IMPROVEMENTS

The first problem is easily overcome by swapping the two points before performing any calculations.

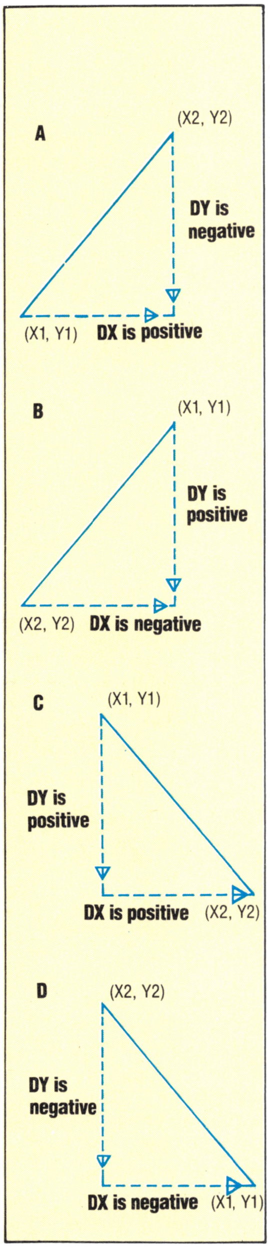
To draw a continuous line we need to increment the value of the X co-ordinate and calculate Y if the gradient is less than one, and increment Y and calculate X for gradients greater than one. Splitting the problem according to this condition (G>1?) allows us to avoid having to use division and double byte two's complement arithmetic.

We've already outlined the plotting procedure for those cases where the gradient is less than one; let's consider what must be done when the gradient exceeds one. Let DX=X2-X1 and DY=Y2-Y1. We can easily determine whether or not the gradient exceeds one by comparing the values of DX and DY. If DX is less than DY, then the gradient is greater than one.

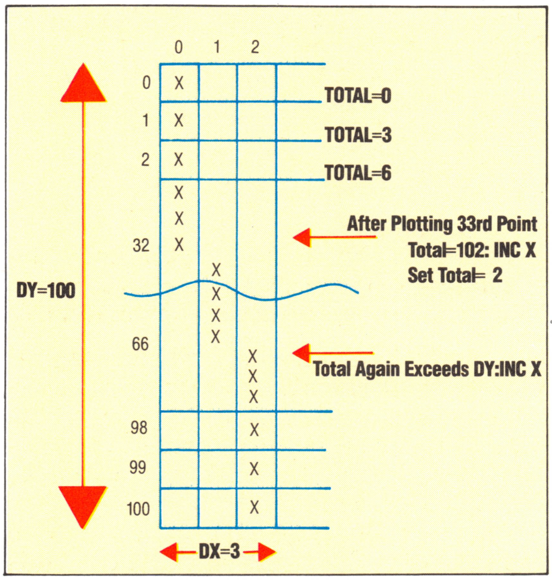
If we wish to avoid negative values of DX and DY, the procedure is more complex. We must consider four possible cases where the gradient is greater than one; these are shown in the diagram. In our program, we must first determine exactly which case we have, and then take the corresponding course of action (assuming that we wish to keep DX positive):

Case a) Recalculate DY as Y1-Y2. Starting at (X1,Y1), we decrement Y until Y2 is reached.
Case b) Swap points and restart.
Case c) Start at (X1,Y1) and increment Y until Y2 is reached.
Case d) Swap points and restart.

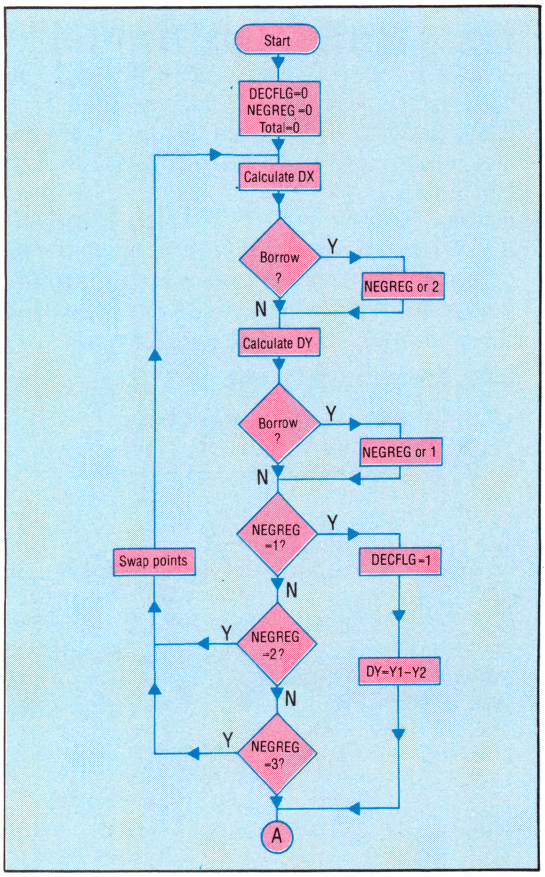
At the beginning of our program we shall calculate DX and DY. Bits within a certain location can be set accordingly if the calculation of DX or DY



Gradient One In Four
When the absolute value of the gradient, (Y2-Y1)/(X2-X1), is greater than one, it is convenient to keep DX positive, by using one of these algorithms:
A) Recalculate -(DY) and decrement Y from (X1,Y1) until Y2 is reached.
B) Swap (X1, Y1) with (X2,Y2) and recalculate
C) Increment Y from (X1, Y1) until Y2 is reached.
D) Swap (X1,Y1) with (X2,Y2) and recalculate



Plot Lines
In the diagram on the left, we show how a horizontal or vertical line can be plotted exactly in a field of discrete points. However, a line at any other angle must be plotted as a series of steps. The line shown here joins the points (0,0) and ((2,100) by three equal vertical steps



Control Lines
The flowchart on the right shows Linesub's method of handling the four cases when the gradient is greater than one. The variables TOTAL and DECFLG are set for use by the routine in actually plotting the flowchart below. Plotting starts at the point (X1,Y1), and, since DX is kept positive, decrements or increments Y according to the state of DECFLG

yields a negative value. Bit 0 of this register (which we will call NEGREG) flags the fact that DY is negative; bit 1 flags DX as being negative. After these calculations, NEGREG can be tested to determine which of the four cases apply:

Case	a	b	c	d
Value of NEGREG	1	2	0	3
Binary Equivalent	01	10	00	11

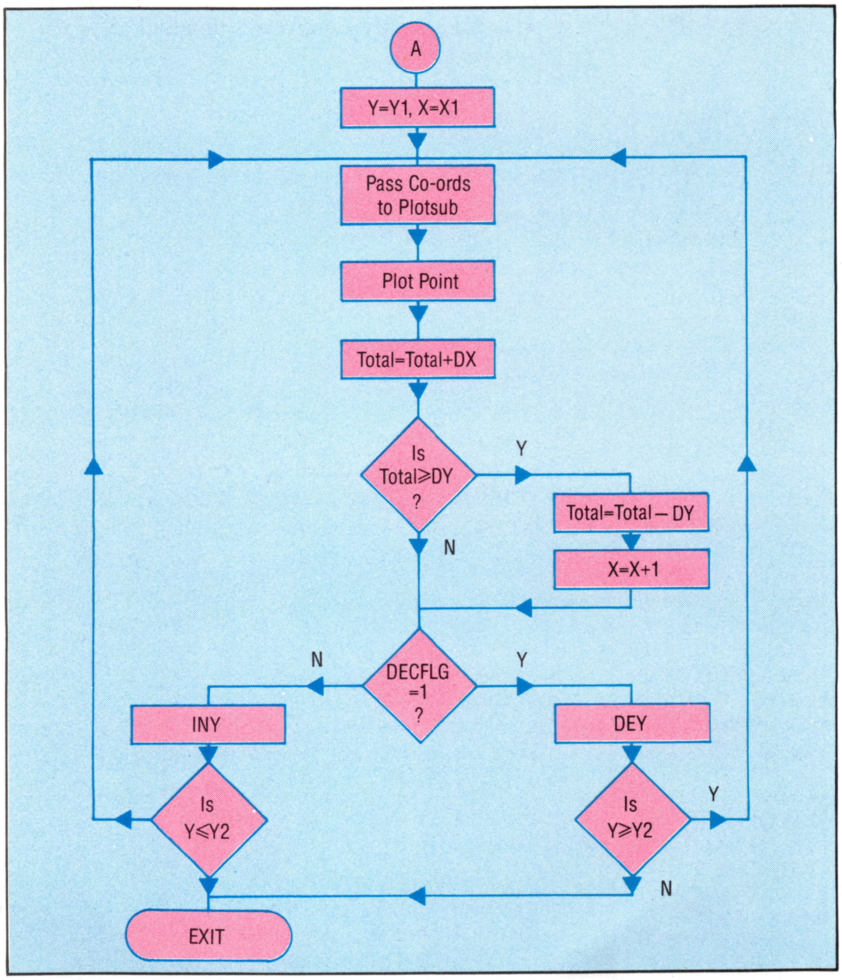
Whichever case applies, we must eventually plot the line using a loop, starting at (X1,Y1), decrementing or incrementing Y as appropriate, and calculating the corresponding value of X before using Plotsub to plot the point. When Y reaches Y2, then the loop can be terminated.

To obtain an expression for calculating X from Y, we must rearrange our original expression once again:

$$(Y-Y1)/(X-X1) = G$$
$$1/(X-X1) = G/(Y-Y1)$$
$$(X-X1) = (Y-Y1)/G$$
$$X = X1 + (Y-Y1)/G$$

and thus, as (Y-Y1) equals one, the expression is: $X = X1 + 1/G$. As $G = DY/DX$, then this expression becomes $X = X1 + DX/DY$. However, if the gradient is greater than one, then DY is always greater than DX; hence, the result of the division DX/DY is always 0, remainder DX. If we keep a running total of the remainders while going through the plotting loop, we should only increment X each time the sum of the remainders exceeds DY. The running total will then be reset.

This may sound a little complex, but we can see the method in operation by considering the example given, where (X1, Y1)=(0,0) and (X2, Y2)=(2,100). The line is displayed as a series of three vertical bars. This is a very steep line; shallower lines would be made up of a greater number of shorter bars until, when the gradient was exactly one, each bar would be only one pixel long, giving the appearance of a perfect diagonal.



Using The Routine From BASIC

In order to use the Linesub routine from BASIC, you must first load Linesub and Plotsub as shown in the BASIC Demonstration Program. We have called our final object code programs PLOTSUB.HEX and LINESUB.HEX. After loading the machine code routines, this short demonstration program asks the user for the co-ordinates of two points, tests the gradient and, if it is not less than one, accesses Plotsub to clear the HIRES screen and set the colour. The subroutine at line 2000 converts the X co-ordinates into Lobyte and Hibyte, before POKEing these values to locations set aside in Linesub for the endpoints of the line. The subroutine at line 3000 prints the values of all the locations used for variables by Linesub. This can be a useful method of debugging machine code programs.

As an alternative to typing in the source code for Linesub and assembling it into machine code, the Machine Code Loader program enters the same program into memory by READing a series of DATA values and POKEing them into memory. Type in this program and RUN it to load Linesub into memory.

BASIC Demonstration Program

```
10 REM ********************
12 REM **  LINESUB 64    **
13 REM ********************
14 :
15 DN=8:REM FOR CASSETTE SET DN=1
20 IF A=0 THENA=1:LOAD"PLOTSUB.HEX",DN,1
30 IF A=1 THENA=2:LOAD"LINESUB.HEX",DN,1
50 INPUT"FIRST POINT";X1,Y1
60 INPUT"SECOND POINT";X2,Y2
70 GOSUB1000: REM SET HIRES MODE
80 GOSUB2000: REM LINESUB
90 GETA$:IFA$=""THEN 90
95 IF A$=" "THEN200
100 REM **** RESET SCREEN ****
110 POKE49408,0:SYS 49422
120 GOSUB3000
125 GETA$:IFA$=""THEN125
127 GOTO50
140 :
200 REM **** DRAW TRIANGLE ****
205 XA=30:YA=10:XB=310:YB=98
210 XC=90:YC=180
220 GOSUB1000
230 X1=XA:Y1=YA:X2=XB:Y2=YB:GOSUB2000
240 X1=XC:Y1=YC:GOSUB2000
250 X2=XA:Y2=YA:GOSUB2000
255 GETA$:IFA$="" THEN 255
260 REM **** RESET SCREEN ****
270 POKE49408,0:SYS 49422
275 PRINTCHR$(147)
280 END
290 :
1000 REM **** SET HIRES ****
1010 POKE49408,1:POKE49409,1
1015 POKE49410,1
1020 SYS 49422
1030 RETURN
1040 :
2000 REM **** ENTER LINESUB ****
2010 MHI=INT(X1/256):MLO=X1-256*MHI
2020 NHI=INT(X2/256):NLO=X2-256*NHI
2030 POKE49920,MLO:POKE49921,MHI
2040 POKE49922,NLO:POKE49923,NHI
2050 POKE49924,Y1:POKE49925,Y2
2060 SYS 49934
2070 RETURN
2080 :
3000 REM **** PRINT VALUES ****
3001 RESTORE
3002 PRINTCHR$(147):REM CLEAR SCREEN
3005 FORI=0TO13
3010 READA$
3020 PRINTA$,PEEK(49920+I)
3030 NEXT I
3040 DATA X1LO,XIHI,X2LO,X2HI,Y1,Y2,DXLO
3050 DATA DXHI,DY,TEMP,TOTLO,TOTHI,NEGREG,DECFLG
3060 RETURN
```

Machine Code Loader

```
10 FOR I=49934 TO 50371
20 READA:CC=CC+A
30 POKEI,A:NEXT
50 READA:IFCC<>A THEN PRINT"CHECKSUM
ERROR":END
100 DATA72,138,72,152,72,169,0,141,13
110 DATA195,141,12,195,141,10,195,141
120 DATA11,195,173,2,195,56,237,0,195
130 DATA141,6,195,173,3,195,237,1,195
140 DATA141,7,195,16,8,173,12,195,9,2
150 DATA141,12,195,173,5,195,56,237,4
160 DATA195,141,8,195,176,8,173,12,195
170 DATA9,1,141,12,195,173,12,195,201
180 DATA1,240,20,201,2,208,6,32,140
190 DATA196,76,19,195,201,3,208,19,32
200 DATA140,196,76,19,195,173,4,195,56
210 DATA237,5,195,141,8,195,238,13,195
220 DATA173,6,195,24,105,1,141,6,195
230 DATA173,7,195,105,0,141,7,195,238
240 DATA8,195,173,4,195,168,173,7,195
250 DATA201,1,240,115,173,6,195,205,8
260 DATA195,176,107,173,0,195,141,3
270 DATA193,173,1,195,141,4,193,152
280 DATA141,5,193,32,131,193,173,10
290 DATA195,24,109,6,195,176,8,141,10
300 DATA195,205,8,195,144,24,56,237,8
310 DATA195,141,10,195,173,0,195,24
320 DATA105,1,141,0,195,173,1,195,105
330 DATA0,141,1,195,173,13,195,201,1
340 DATA208,31,136,204,5,195,240,3,76
350 DATA161,195,152,141,5,193,173,0
360 DATA195,141,3,193,173,1,195,141,4
370 DATA193,32,131,193,76,134,196,200
380 DATA204,5,195,144,152,76,134,196
390 DATA173,0,195,141,3,193,173,1,195
400 DATA141,4,193,152,141,5,193,32,131
410 DATA193,173,10,195,24,109,8,195
420 DATA141,10,195,173,11,195,105,0
430 DATA141,11,195,173,10,195,56,237,6
440 DATA195,141,10,195,173,11,195,237
450 DATA7,195,141,11,195,48,15,173,13
460 DATA195,201,1,240,4,200,76,104,196
470 DATA136,76,104,196,173,10,195,24
480 DATA109,6,195,141,10,195,173,11
490 DATA195,109,7,195,141,11,195,173,0
500 DATA195,24,105,1,141,0,195,173,1
510 DATA195,105,0,141,1,195,205,3,195
520 DATA208,142,173,0,195,205,2,195
530 DATA208,134,104,168,104,170,104,96
540 DATA173,2,195,141,9,195,173,0,195
550 DATA141,2,195,173,9,195,141,0,195
560 DATA173,3,195,141,9,195,173,1,195
570 DATA141,3,195,173,9,195,141,1,195
580 DATA173,5,195,141,9,195,173,4,195
590 DATA141,5,195,173,9,195,141,4,195
600 DATA96,230
610 DATA50794:REM*CHECKSUM*
```

Assembly Listing

```
                ;++++++++++++++++++++
                ;++++++++++++++++++++
                ;++                ++
                ;++   LINESUB 64   ++
                ;++                ++
                ;++++++++++++++++++++
                ;++++++++++++++++++++
                ;
                PLTSUB = $C183
                XLO    = $C103
                XHI    = $C104
                YLO    = $C105
                *      = $C300
                ;
                ;++++ LINESUB VARIABLES ++++
                ;
                X1LO   *=*+1
                X1HI   *=*+1
                X2LO   *=*+1
                X2HI   *=*+1
                Y1     *=*+1
                Y2     *=*+1
                DXLO   *=*+1
                DXHI   *=*+1
                DY     *=*+1
                TEMP   *=*+1
                TOTLO  *=*+1
                TOTHI  *=*+1
                NEGREG *=*+1
                DECFLG *=*+1
                ;
                ;++++ PUSH REGISTERS ONTO STACK ++++
                ;
48                      PHA
8A                      TXA
48                      PHA
98                      TYA
48                      PHA
                ;
                ;++++ INITIALISE VARIABLES ++++
                ;
                START
A9 00                   LDA #$00
8D 0D C3                STA DECFLG
8D 0C C3                STA NEGREG
8D 0A C3                STA TOTLO
8D 0B C3                STA TOTHI
                ;
                ;++++ CALCULATE DX ++++
AD 02 C3                LDA X2LO
38                      SEC
ED 00 C3                SBC X1LO
8D 06 C3                STA DXLO
AD 03 C3                LDA X2HI
4C 86 C4                JMP FINISH
                ;
                NODEY
C8                      INY
CC 05 C3                CPY Y2
90 98                   BCC PLOT
                ;
4C 86 C4                JMP FINISH
                ;++++ GRADIENTS LESS THAN ONE ++++
                ;
                GLESS
AD 00 C3                LDA X1LO
8D 03 C1                STA XLO
AD 01 C3                LDA X1HI
8D 04 C1                STA XHI
98                      TYA
8D 05 C1                STA YLO
20 83 C1                JSR PLTSUB
                ;
AD 0A C3                LDA TOTLO
18                      CLC
6D 08 C3                ADC DY          ; TOTAL=TOTAL+DY
8D 0A C3                STA TOTLO
AD 0B C3                LDA TOTHI
69 00                   ADC #$00        ; CATCH CARRY
8D 0B C3                STA TOTHI
                ;
AD 0A C3                LDA TOTLO
38                      SEC
ED 06 C3                SBC DXLO
8D 0A C3                STA TOTLO  :TOTAL=TOTAL-DX
AD 0B C3                LDA TOTHI
ED 07 C3                SBC DXHI
8D 0B C3                STA TOTHI
30 0F                   BMI ADD         ;NEGATIVE RESULT
                ;
                ;
AD 0D C3                LDA DECFLG
C9 01                   CMP #$01
F0 04                   BEQ DECY
C8                      INY
4C 68 C4                JMP PAST
                DECY
88                      DEY
4C 68 C4                JMP PAST
                ;
                ADD
AD 0A C3                LDA TOTLO
18                      CLC
6D 06 C3                ADC DXLO        ; ADD BACK DX
8D 0A C3                STA TOTLO
AD 0B C3                LDA TOTHI
6D 07 C3                ADC DXHI
8D 0B C3                STA TOTHI
ED 01 C3                SBC X1HI
8D 07 C3                STA DXHI
10 08                   BPL NBRWDX
AD 0C C3                LDA NEGREG
09 02                   ORA #$02        ;FLAG DX NEGATIVE
8D 0C C3                STA NEGREG
                ;
                ;++++ CALCULATE DY ++++
                ;
                NBRWDX
AD 05 C3                LDA Y2
38                      SEC
ED 04 C3                SBC Y1
8D 08 C3                STA DY
B0 08                   BCS NBRWDY
AD 0C C3                LDA NEGREG
09 01                   ORA #$01        ;FLAG DY NEGATIVE
8D 0C C3                STA NEGREG
                ;
                NBRWDY
                ;++++ TEST STATE OF NEGREG ++++
                ;
AD 0C C3                LDA NEGREG
C9 01                   CMP #$01
F0 14                   BEQ NEG1
                ;
C9 02                   CMP #$02
D0 06                   BNE FALSE
20 8C C4                JSR SWOP
4C 13 C3                JMP START
                ;
                FALSE
C9 03                   CMP #$03
D0 13                   BNE NEG0
20 8C C4                JSR SWOP
4C 13 C3                JMP START
                ;
                NEG1
AD 04 C3                LDA Y1
38                      SEC
ED 05 C3                SBC Y2     :RECALCULATE DY
8D 08 C3                STA DY            ;AS Y1-Y2
EE 0D C3                INC DECFLG;SET DECFLG
                ;
                NEG0
AD 06 C3                LDA DXLO
18                      CLC
69 01                   ADC #$01        ; ADD 1 TO DX
8D 06 C3                STA DXLO
AD 07 C3                LDA DXHI
69 00                   ADC #$00        ; CATCH CARRY
8D 07 C3                STA DXHI
EE 08 C3                INC DY          ; ADD ONE TO DY
AD 04 C3                LDA Y1
A8                      TAY             ;SET Y=Y1
                ;
                ;++++ TEST FOR DX >=DY ++++
                ;
AD 07 C3                LDA DXHI
C9 01                   CMP #$01
F0 73                   BEQ GLESS
AD 06 C3                LDA DXLO
CD 08 C3                CMP DY
B0 6B                   BCS GLESS
                ;
                ;++++ PLOT POINT ++++
                ;
                PLOT
AD 00 C3                LDA X1LO
8D 03 C1                STA XLO         ;PASS LOBYTE OF X
AD 01 C3                LDA X1HI
8D 04 C1                STA XHI         ;PASS HIBYTE OF X
98                      TYA
8D 05 C1                STA YLO         ;PASS Y
20 83 C1                JSR PLTSUB
AD 0A C3                LDA TOTLO
18                      CLC
6D 06 C3                ADC DXLO        ;TOTAL=TOTAL+DX
B0 08                   BCS DOINX       ;SUM EXCEEDS 1 BYTE?
8D 0A C3                STA TOTLO
CD 08 C3                CMP DY
90 18                   BCC NOINX       ;TOTAL REACHED DY?
                DOINX
38                      SEC
ED 08 C3                SBC DY          ;TOTAL=TOTAL-DY
8D 0A C3                STA TOTLO
AD 00 C3                LDA X1LO
18                      CLC
69 01                   ADC #$01
8D 00 C3                STA X1LO        ; INCREMENT X1
AD 01 C3                LDA X1HI
69 00                   ADC #$00
8D 01 C3                STA X1HI
                ;
                NOINX
AD 0D C3                LDA DECFLG
C9 01                   CMP #$01
D0 1F                   BNE NODEY       ;DECFLG=1?
88                      DEY
CC 05 C3                CPY Y2
F0 03                   BEQ FINAL       ; IF Y=Y2 PLOT LAST
4C A1 C3                JMP PLOT
                FINAL                   ;PLOT LAST POINT
98                      TYA
8D 05 C1                STA YLO
AD 00 C3                LDA X1LO
8D 03 C1                STA XLO
AD 01 C3                LDA X1HI
8D 04 C1                STA XHI
20 83 C1                JSR PLTSUB
                ;
                PAST
AD 00 C3                LDA X1LO
18                      CLC
69 01                   ADC #$01
8D 00 C3                STA X1LO        ; INCREMENT X
AD 01 C3                LDA X1HI
69 00                   ADC #$00
8D 01 C3                STA X1HI        ; CATCH CARRY
                ;
CD 03 C3                CMP X2HI
D0 8E                   BNE GLESS
AD 00 C3                LDA X1LO        ; HAS X REACHED X2?
CD 02 C3                CMP X2LO
D0 86                   BNE GLESS
                ;
                ;++++ PULL REGISTERS OFF STACK ++++
                ;
                FINISH
68                      PLA
A8                      TAY
68                      PLA
AA                      TAX
68                      PLA
                ;
60                      RTS
                ;
                ;++++ END OF PROGRAM ++++
                ;
                ;++++ SWOP POINTS SUBROUTINE ++++
                ;
                SWOP
AD 02 C3                LDA X2LO
8D 09 C3                STA TEMP
AD 00 C3                LDA X1LO
8D 02 C3                STA X2LO
AD 09 C3                LDA TEMP        ;SWOP LOBYTES OF X
8D 00 C3                STA X1LO
                ;
AD 03 C3                LDA X2HI
8D 09 C3                STA TEMP
AD 01 C3                LDA X1HI
8D 03 C3                STA X2HI
AD 09 C3                LDA TEMP        ;SWOP HIBYTES OF X
8D 01 C3                STA X1HI
                ;
AD 05 C3                LDA Y2
8D 09 C3                STA TEMP
AD 04 C3                LDA Y1
8D 05 C3                STA Y2
AD 09 C3                LDA TEMP        ; SWOP Y
8D 04 C3                STA Y1
                ;
60                      RTS
                ;
                ;++++ END OF SUBROUTINE ++++
```

Linesub Machine Code Routine

The machine code routine draws lines with gradients greater than, equal to, or less than one. There is no problem with vertical lines as, using the method we have described, there is no division by zero required. For lines with gradients less than one, the same principles apply as those outlined. All that is required is to insert a test to determine whether or not DX is greater than DY (indicating a gradient less than one), and branching accordingly.

■ The routine uses two bytes for each X co-ordinate since X may exceed 255. Y however, cannot exceed 199 and can therefore be held in a single byte.

■ The locations used by Plotsub to hold the co-ordinates to be plotted are assigned at the beginning of the Assembly listing. The start address of Plotsub is also assigned.

■ Bits 1 and 0 of NEGREG are set according to the results of calculating DX and DY. In order to access each bit separately, without affecting the other, use is made of the logical ORA instruction.

■ As the Y co-ordinate can be held in a single byte, the Y register is used to hold its value during plotting. The register values are not corrupted by Plotsub.

N.B. In order to ensure that Plotsub can be used with Linesub, the following change should be made to your source code listing (on page 339). A branch instruction must be inserted between SBC REMX and TAX.

The equivalent change can be made to the BASIC Demonstration program by inserting these lines:

```
5010 FOR I=HRSFLG TO HRSFLG+314:READ A
5380 DATA 169,7,237,8,193,240,6,170,14,13,193
5410 DATA 38944:REM*CHECKSUM*
```

ARTICULATE VOICE

Artic Computing is a classic success story. It was founded in 1981 with £20 of pocket money by an 18-year-old schoolboy called Richard Turner. Since then it has developed into a software company with an annual turnover of around £750,000, and plans for worldwide expansion.

Richard Turner started writing software in 1980. The first games he wrote were Battleships and Star Trek for the ZX80 computer. He chose to write strategy games instead of the more popular arcade-style games owing to the limitations of the machine. As he explains: 'The ZX80 used to clear the screen every time something moved, so the only games you could do were thinking games and not arcade games, which only really came in with the Spectrum.'

His first big success was with the game ZX Chess, which he launched at the first ZX Microfair in the summer of 1981. Turner's resources were pushed to the limit: 'The night before, we were still copying cassettes using seven ZX81s and putting them in plastic bags with instructions we'd run off the school photocopier.' ZX Chess was a great success, and Turner claims to have taken £1,500 at the fair.

Artic Computing became a limited company later that summer but had to take a back seat when Turner accepted a sponsorship from the Ford Motor Company to study Electrical Engineering at Imperial College, London. His studies lasted only a year, at the end of which he decided to take a year's break to run the company. He never returned to the University.

Artic was originally run from Richard's bedroom in his parents' house in Hull, but as the company's list of software grew to 93 titles, Turner decided that it had to have premises of its own. In June 1983 the company moved to its present offices at Brandesburton, Humberside. The catalogue was streamlined and extra staff were taken on. Artic now employs 15 people, including three telesales staff and five full-time programmers, who are paid a salary plus royalties.

Artic plans to open its own chain of retail outlets throughout the UK. These will be known as Artic Software Stations, and will sell not only Artic's games but also the products of other companies. The first 'station' opened in Acton, West London, in July 1984 and doubles as the company's London headquarters. The significance of the shop is that it is not in a main shopping precinct and is well away from the commercial centre of London's West End. Asked why he chose this particular site, Jeff Raggett, Artic's London marketing manager, replied: 'A high street site would be £300-£400 a week, and this shop is a lot less, so we don't have to sell many cassettes to cover overheads. A lot of people have criticised us, saying we're mad to open shops, but at least we can see what is selling, and can talk to people about what they like about the games.'

Another marketing innovation is Artic's counter units. These units are display boxes that can hold up to 64 cassettes. They are currently being sold to newsagents, allowing people to buy their software locally rather than having to go to the large retailers. Jeff Raggett says that these units are proving extremely successful.

Artic plans to handle its own overseas marketing as much as possible, and the company is currently investigating expansion into Europe. For the North American market, Artic has made a contract with two established software houses, Softsync and the International Publishing Corporation, to distribute each other's products.

Artic's biggest sellers to date have been Bear Bovver (which alone has sold over 40,000 cassettes), Galaxians and Gobbleman. The company has recently launched a new game for the Spectrum called World Cup, which sold over 5,000 copies in three weeks.

Artic Games
A range of the games for which Artic is noted, including the best-selling Bear Bovver and World Cup, a Spectrum version of the popular football game

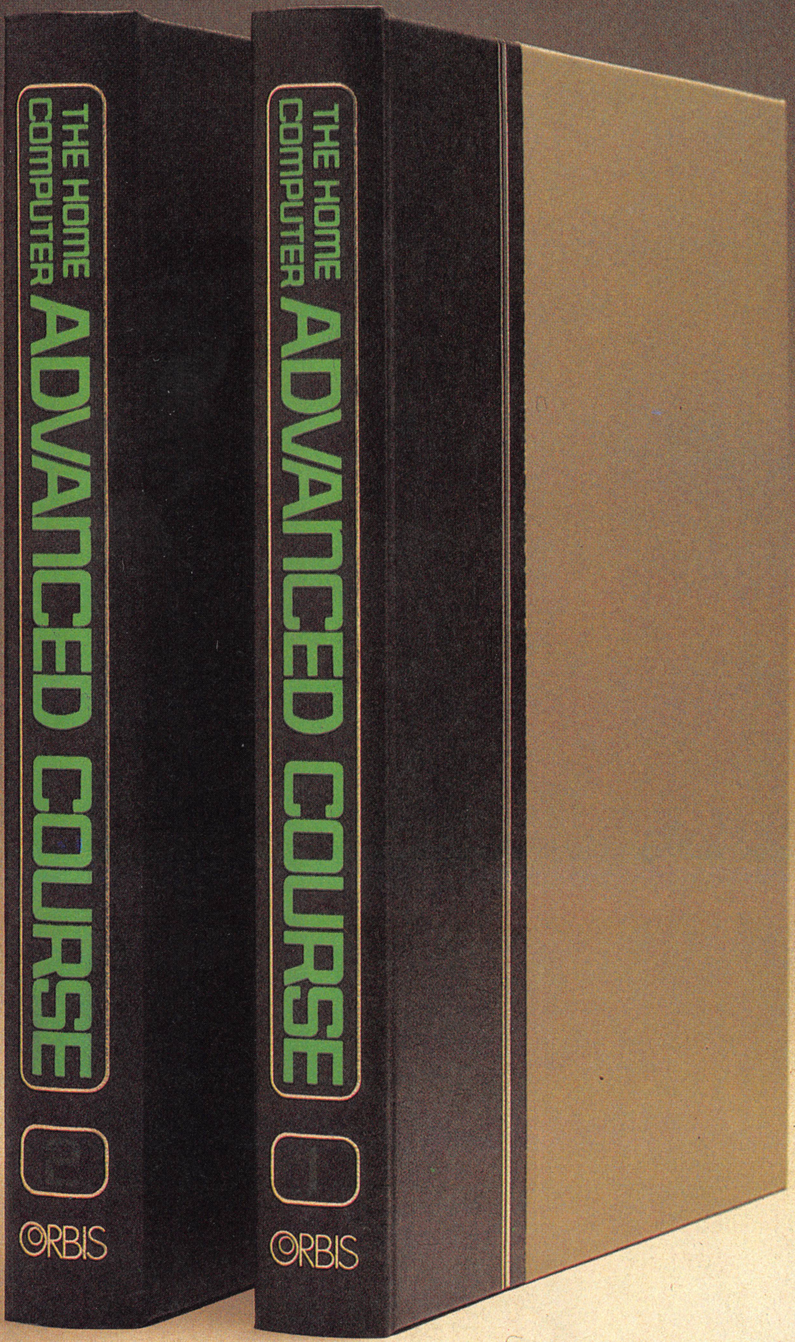
IAN McKINNELL

Richard Turner


Mentathlete
Home computers. Do they send your brain to sleep – or keep your mind on its toes?
At Sinclair, we're in no doubt. To us, a home computer is a mental gym, as important an aid to mental fitness as a set of weights to a body-builder.
Provided, of course, it offers a whole battery of genuine mental challenges.
The Spectrum does just that.
Its education programs turn boring chores into absorbing contests – not learning to spell 'acquiescent', but rescuing a princess from a sorcerer in colour, sound, and movement!
The arcade games would test an all-night arcade freak – they're very fast, very complex, very stimulating.
And the mind-stretchers are truly fiendish. Adventure games that very few people in the world have cracked. Chess to grand master standards. Flight simulation with a cockpit full of instruments operating independently. Genuine 3D computer design.
No other home computer in the world can match the Spectrum challenge – because no other computer has so much software of such outstanding quality to run.
For the Mentathletes of today and tomorrow, the Sinclair Spectrum is gym, apparatus and training schedule, in one neat package. And you can buy one for under £100.
ZX Spectrum
sinclair

THE HOME COMPUTER ADVANCED COURSE

WE HAVE DESIGNED BINDERS SPECIALLY TO KEEP YOUR COPIES OF THE 'ADVANCED COURSE' IN GOOD ORDER.



All you have to do is complete the reply-paid order form opposite – tick the box and post the card today – **no stamp necessary!**

By choosing a standing order, you will be sent the first volume free along with the second binder for £3.95. The invoice for this amount will be with the binder. We will then send you your binders every twelve weeks – as you need them.

Important: This offer is open only whilst stocks last and only one free binder may be sent to each purchaser who places a Standing Order. Please allow 28 days for delivery.
Overseas readers: This free binder offer applies to readers in the UK, Eire and Australia only. Readers in Australia should complete the special loose insert in issue 1 and see additional binder information on the inside front cover. Readers in New Zealand and South Africa and some other countries can obtain binders now. For details please see inside the front cover. Binders may be subject to import duty and/or local tax.
The Orbis Guarantee: If you are not entirely satisfied you may return the binder(s) to us within 14 days and cancel your Standing Order. You are then under no obligation to pay and no further binders will be sent except upon request.

PLACE A STANDING ORDER TODAY.